WORKSHOP e€ (ELECTRONIC MONEY)

MICHAEL NÜSKEN

April 7, 2008

Contents

Distinguish money vs. cash: Cash is a form of money, but not the only one...

For the implementation of electronic money and electronic cash there are a lot of things to understand. From top to bottom, we find that we need

- Protocols, that is, formalized chats, to make the system exchange the desired information at the right place and the right time.
- Signatures (and authentication) to mimic essential properties of real money as unforgeability (of a reasonable strength).
- Number theory to implement these things.
- A programming language to bring everything to life.

All these items are of course only the technical side of the story. The social and economical aspects form an additional field. The reason for enterprises like DigiCash going bankrupt were not bad technical solutions.

1. Properties of € (cash) and e€ (electronic cash)

There are various properties that can be asked of electronic money. A problem is that there are often no standard names, so you find two or three different words for the same thing.

unforgeable This is a basic part of security: it should not be possible to forge a coin.

double spending protected This is a basic part of security: it should not be possible to spend a coin twice, or at least it must be guaranteed that double spending has serious consequences.

Online vs. offline vs. with observer This refers to the payment protocol in an electronic payment system. Some systems require an interaction with the bank at this point, they are called *online*. For example, if you pay with your bank card at the supermarket and enter your PIN then the bank is called for clearance. Other system do not need this, they are offline. Apart from these two extremes there is a further solution: typically the customer gets a card from the bank anyway. So if this card can do some active computation we may use it to carry a further secret such that the card can acknowledge transactions by a signature. Such a card then is called an *observer*. Though the bank is not directly contacted during this kind of payment, I think that the observer has to be considered as part of the bank and thus it is somehow online. Yet, with respect to scalability the system behaves like an offline system, there is no central server that has to deal with every payment.

untraceable, anonymous, privacy protected vs. fair An electronic payment system that does not allow the bank to trace who paid whom which amount is called *anonymous* or *untraceable*, sometimes this property comes along under the heading *privacy*. Bank transfers are *not* anonymous. As we will see later unconditional anonymity might actually not be wanted. Using a trustee, possibly split, one can build systems that allow to trace coins when a court decides that this is necessary. Still, the bank shall not be able to trace a coin unless a double spending occurs.

One might also consider privacy protection with respect to the merchant as a possible tracer but in any case the merchant must know where to send his goods to.

atomic It must be clear to all participants when a coin is transferred. A partial completion of the payment protocol should not cause conflicts.

reusable, transferable This means that a customer can hand over coins to another customer and so on. The coin must only be transferred to the bank after a certain maximal number of transfers. This is similar to the ordinary life of a physical coin.

divisible It might be nice to have coins that can be divided into smaller ones as needed.

account-free vs. -based Electronic money is always account-free, most electronic payment system are account-based.

acceptable Which bank takes your coins?

reliable The system should always work when it is needed.

independent The system should not depend on any physical location. The money can be transferred through computer networks.

2. Public key cryptography

Key words: RSA and its security, simple attacks. Signing with RSA. Group based cryptography, examples, simple attacks. Authentication. (Zero knowledge?)

2.1. Rivest, Shamir & Adleman (1978).

Public key cryptography was developed in the seventies. Only when, shortly after the key exchange protocol by Diffie & Hellman (1976), Rivest, Shamir & Adleman (1978) published the first publicly known asymmetric cryptosystem, now named RSA after their inventors, the raise of public key cryptography began. Before that it was commonly thought to be impossible that two previously unknown parties could exchange secrets in the presence of eavesdroppers. Only symmetric key systems were used until then. The following decades brought more and more electronic means, in particular the Internet. There was and is an increasing need of confidentiality and security.

Curiously, the British secret service discovered public key cryptography already about five years earlier and astonishingly the two schemes they came up with were RSA and the Diffie Hellman key exchange. See Ellis (1987) for an historical account, the original papers are Cocks (1973); Ellis (1970); Williamson (1974, 1976).

RSA is fairly simple. For a start you need two prime numbers p and q. Form their multiple $N = p \cdot q$. If you calculate modulo N then there are only finitely many possible results. Thus if you repeat a specific operation, say multiplying by a fixed number x, the sequence of results $(1, x, x^2, x^3, x^4, \dots)$ must become periodic at some point. Actually $L = (p-1)(q-1)$ is a repetition length (Later, we will prove this!) and the period starts almost immediately: $x = x^{1+L} = x^{1+2L} = \dots$. Thus if $ed = 1$ in $\mathbb{Z}_L$ then $x^{ed} = x^1$ in $\mathbb{Z}_N$. As we will see it is easy to calculate such a pair (e, d) provided one knows the repetition length L.

Suppose Alice has computed $N = p \cdot q$ and two numbers e and d as above. Now if she wants to obtain a message from Bob, she sends him her public key (N, e). Bob translates the messages into a number $x \in \mathbb{Z}_N$ (for example by writing it using ASCII characters and concatenating the bits to a long integer) and calculates $y \leftarrow x^e$ in $\mathbb{N}$. (This is also very easy, as we will see.) When Alice gets y she computes $z \leftarrow y^d$ in $\mathbb{Z}_N$. Now, $z = y^d = (x^e)^d = x^{ed} = x$ in $\mathbb{Z}_N$ and so Alice can read the message. Eve, listening to the entire conversation including the explanations about the system, cannot decode the message: She knows N, e and y. Though this identifies the message, she needs to solve the equation $x^e = y$ in $\mathbf{Z}_N$. This is considered to be a difficult problem, even if Eve would content herself with guessing a single bit of x.

There are several important questions:

- ○ Correctness: Does the protocol fulfill its demands? (And why? Note that the 'why' is important! It enables us to look for generalizations as well as for attacks.)
- ○ Efficiency: How fast can the necessary operations be performed?
- ○ Security: How fast can the best possible attack break the system? Or more modestly: How fast are the best known attacks?

2.1.1. Correctness.

We have already seen that $z = y^d = x^{ed} = x$ in $\mathbb{Z}_N$ since $ed = 1$ in $\mathbb{Z}_L$. However, the value for the repetition length still has to be proved. We postpone that to Section 3.

2.1.2. Efficiency. To answer the question about the timing we have to consider the operations to be performed. They are

- Finding primes p and q. This splits into the subtasks
  - Choose a random $n/2$-bit number.
  - Test whether it is prime.
  - Repeat until you find a prime.

  Choosing random numbers may be costly depending on the desired security. If an attacker can guess the results of the random number generation she will know all we do. Even partial attacks are dangerous.

  Testing primality can be performed in deterministic polynomial time. This is known only since Agrawal, Kayal & Saxena (2003) (mostly cited as AKS). Heuristically this algorithm runs in time $\mathcal{O}^{\sim}(n^6)$ after embedding several enhancements. In practice, this is much too slow. But fast probabilistic algorithms run in time $\mathcal{O}^{\sim}(kn^2)$ with fast and $\mathcal{O}(kn^3)$ with classical arithmetic and return a wrong answer with probability at most 2^{-k}. This is only by a factor kn slower than the time to multiply two numbers of similar size.

  This has to be repeated until a prime is found. The probability that an n-bit number is a prime is about $\frac{1}{n \ln 2}$. Thus an expected number of $\mathcal{O}^{\sim}(n)$ repetitions leads to a prime number. (Here, some care is necessary, in particular, if the prime test is only a probabilistic one.)

  Of course, we have to do all this twice but we ignore constant factors anyway.

- Compute $N = p \cdot q$ and $L = (p-1)(q-1)$. Multiplying n-bit numbers can be performed in time $\mathcal{O}(n^2)$ with classical arithmetic or in time $\mathcal{O}(n \log n \operatorname{loglog} n)$ by Schönhage & Strassen (1971). The simple and practical Karatsuba algorithm achieves $\mathcal{O}(n^{\log_2 3})$. More information about the arithmetic of integers can be found in von zur Gathen & Gerhard (2003).

- Find e and d. This splits into the subtasks
  - Choose a random n-bit number e.
  - Decide whether d exists and compute it.
  - Repeat until d is found.

  The first task is as above. The two aspects of the second can be done at once by the Extended Euclidean Algorithm. It needs $\mathcal{O}(n^2)$ operations with classical arithmetic. A clever fast implementation achieves the desired result (but not the entire EEA!) with $\mathcal{O}(n \log^2 n \operatorname{loglog} n)$ operations.

- Compute x^e in $\mathbb{Z}_N$. This is actually the simplest thing: Just start with 1 and multiply by x in $\mathbb{Z}_N$ until you reach x^e. Of course, the multiplication should be performed in $\mathbb{Z}_N$ in order to keep the memory requirements small. This only takes e multiplications of n-bit numbers. … — Sorry, I was kidding: e itself is an n-bit number, too, and doing 2^n multiplications may take longer than we live even with the best of all computers that we can imagine. OK, how to do better? The answer is *repeated squaring*: for example, computing x^{256} is easy by simply squaring eight times. We only need 8 instead of 256 multiplications. To compute any power of x write e in binary, for example, if $e = 1011101_2$ we compute x, x^{10}, x^{100}, x^{101}, x^{1010}, x^{1011}, x^{10110}, x^{10111}, x^{101110}, $x^{1011100}$, $x^{1011101}$. In other words: either we square the last result which attaches a zero to the end of the binary representation of the reached exponent or we multiply the last result by x which adds 1 to the exponent. This we use in order to build up the desired exponent. That takes at most $2(n-1)$ multiplications. We need $n-1$ squarings and one multiplication for each 1 in the binary representation for e. Slight improvements are possible but you will never get along with less than $n-1$ squarings. Now, the time here is: $\mathcal{O}(n^3)$ with classical and $\mathcal{O}^{\sim}(n^2)$ with fast arithmetic.

For short, all operations can be performed in polynomial time. In practice, choosing random numbers and verifying primality are the most time consuming parts. Luckily, these occur only in the key generation procedure.

To understand all this properly we need some knowledge about modular arithmetic and its implementation. In particular, multiplication of large integers, division with remainder and, most prominently, the Extended Euclidean Algorithm.

There are some possibilities to speed up things, in particular with specially tailored hardware:

- ○ Using a special e, which is small and has only few ones in its binary representation, can make exponentiation slightly faster. But be aware of possible new attacks!
- ○ Using the Chinese Remainder Theorem 3.1 can speed up the decryption by using $N = p \cdot q$. Yet, this may be a bad idea if the calculations are performed in a, say, stolen smart card since it involves more secret data than necessary.

2.1.3. Security. Let's consider this in more detail: Eve knows N, e and y. She would like to know the plain text x or anything that gives her knowledge about it. To break the system completely she could try to

(A) factor the modulus N and find $N = p \cdot q$.

(B) find the repetition length L.

(C) find the decryption exponent d such that $(x^e)^d = x$ for all $x \in \mathbb{Z}_N$.

(D) find the plain text x such that $y = x^e$ in $\mathbb{Z}_N$.

Clearly, (A) is equivalent to (B) and implies (C) which in turn implies (D). [Note that $(x-p)(x-q) = x^2 - (N+1-L)x + N$, so given L computing p and q simply amounts to solve a known quadratic equation.] Though one can obtain (A) from a suitable form of (C), it is an open problem whether a form of (D) implies one of the other items. After all, not only these complete breakings are threats to RSA. If an attacker would be able to guess a bit of x with probability significantly larger than 50%, then this is already considered to be a breaking.

There are many attacks on RSA, or better, false usage of RSA.

1. Chosen cipher text attacks.

   Suppose Alice uses the same RSA key pair for encryption and signing. (Signing with RSA is done as follows: The signer decrypts the message or its hash value. This serves as a signature. It can be generated only with the knowledge of the secret key, so only the signer can produce it. It can be easily verified using the public key.)

   - ○ Eve collects a message $y = x^e$ in $\mathbb{Z}_N$ for Alice. To decode it Eve computes $y' = r^e y$ with some random $r \in \mathbb{Z}_N^\times$ and makes Alice sign y': she gets $z' = r^{ed} y^d = r y^d = rx$. Now she can easily divide by r and obtain $x = z'/r$.

   Trent is a public computer notary. He signs any document given to him to grant its existence at the usage period during the present key life time.

   - ○ Mallory wants Trent to sign a criminal document x. Mallory makes Trent sign $x' = xr^e$ instead and obtains $y' = x^d r^{ed} = x^d r$ from which he easily obtains the signature $y = y'/r = x^d$ to x.
   - ○ Mallory can also write $x = x_1 x_2$ and make Trent sign both factors: $y_i = x_i^d$. Then the wanted signature is $x^d = x_1^d x_2^d$.

   There are further attacks on encrypting and signing with the same RSA key pair. See Schneier (1996), section 19.3, page 473.

2. Common modulus attacks.

   For broadcasting purposes Bob and Carol have the public keys (N, e_1) and (N, e_2). To broadcast x Alice computes x^{e_1} and x^{e_2} in $\mathbb{Z}_N$ and sends them. Eve now simply finds s and t such that $1 = se_1 + te_2$ and computes $x = x^1 = (x^{e_1})^s (x^{e_2})^t$.

3. Low encryption exponent, e less than number of recipients.

   If, for a broadcast say, you encrypt the same messages x with the same e and e different values for N then an attacker can recover x easily. Namely, if you know x^e modulo N_1, N_2, $\ldots$, N_e then by the Chinese Remainder Theorem 3.1 you also know x^e modulo the product $M := \prod_{1 \leq i \leq e} N_i$ of all these moduli. Since M is larger than x^e, we have the integer x^e and extracting its e-th root is simple. This attack can be generalized to linearly dependent (instead of equal) messages.

4. Low decryption exponent, $d < N^{1/4}$.

   There is an attack by Michael Wiener that recovers such a small d by approximating the rational number $\frac{k}{d} = \frac{e}{L} - \frac{1}{dL}$ with the known fraction $\frac{e}{N}$. Using continued fractions we can recover d provided it is small enough.

3. Number theory

In this section we deal with basics on modular arithmetic, group theory and finite fields. It should cover:

- ○ modulo calculations, $\mathbb{Z}_N$,
- ○ the Chinese Remainder Theorem 3.1,
- ○ EEA,
- ○ inverting elements modulo some integer, $\mathbb{Z}_N^\times$,
- ○ the Euler totient function,
- ○ finite groups, Abelian,
- ○ Lagrange's Theorem 3.4, Euler's Theorem 3.5, Fermat's Little Theorem 3.6,
- ○* element order, cyclic, group generator,
- ○ repeated squaring,
- ○* element order test,
- ○ discrete logarithms,
- ○* finite fields, in particular their multiplicative group,
- ○* an additional example: elliptic curves. (An overview is in Kou (2003), section 2.4.3, page 20-22.)

Not all of these topics are worked out in detail.

As promised, we are going to prove that RSA works, that is $x^{ed} = x$ for $x \in \mathbb{Z}_N$ if $ed = 1$ in $\mathbb{Z}_L$. To do so we first need to clarify where our numbers live and how they behave.

3.1. The ring $\mathbb{Z}_N$ of integers modulo N. We introduce the ring $\mathbb{Z}_N$, called integers modulo N, as a set of N elements with an addition and a multiplication, a zero element and a one element. Typically, the implementation uses the set $\mathbb{N}_{<N}$ of non-negative integers less than N as a set and the addition and multiplication are simply the addition and multiplication of integers with results reduced modulo N. (Alternatively a symmetrical set is often used, $\left\{x \in \mathbb{Z} \,\middle|\, -\frac{N}{2} \leq x < \frac{N}{2}\right\}$.) It can be understood as a class in an object oriented language, so $4 \cdot 5 = 6$ in $\mathbb{Z}_7$ is immediately clear, the multiplication is performed as the class demands (and not as for integers). That $\mathbb{Z}_N$ is a *ring* means (don't PANIC, PAN, D):

P+,P· It is **p**roperly defined, that is, it consists of a set S, a zero $0 \in S$, an addition $+\colon S \times S \to S$, a one $1 \in S$ and a multiplication $\cdot\colon S \times S \to S$.

A+,A· Both operations are **a**ssociative, $(a+b)+c = a+(b+c)$, $(ab)c = a(bc)$.

N+,N· Both operations have a **n**eutral element, $a+0 = a = 0+a$, $a \cdot 1 = a = 1 \cdot a$.

I+ The addition has **i**nverses: $a+x = 0 = x+a$ is always solvable. (And thus uniquely: if $a+x=0$ and $y+a=0$ then $x = 0+x = (y+a)+x = y+(a+x) = y+0 = y$. You may also implement the operation $a \mapsto x$ as $-$.)

C+ The addition is **c**ommutative: $a+b = b+a$.

D Addition **d**istributes over multiplication: $(a+b)c = ac+bc$, $a(b+c) = ab+ac$.

A lot of rings, including our example, are called commutative. That means:

C· The multiplication is **c**ommutative: $a \cdot b = b \cdot a$.

If additionally the multiplication has inverses for all non-zero elements, I·, then the ring is even a *field*. Actually, in case p is a prime then $\mathbb{Z}_p$ is a field as we will see shortly. Still, if not all non-zero elements do have a multiplicative inverse, some may have. In any case the question arises how to compute them in our ring $\mathbb{Z}_N$. Actually, for RSA we will need that for computing $d \in \mathbb{Z}_L$. But we are still on the way to prove that RSA works, so keep that question in mind for later.

3.2. The Chinese Remainder Theorem. Coming from RSA, we want to prove that L is a repetition length for the multiplication with $x \in \mathbb{Z}_N$, that is, $x^{1+L} = x$ for all $x \in \mathbb{Z}_N$. The nice way to prove that is to split the problem into two halves, one for each factor of $N = p \cdot q$. This is done by the famous Chinese Remainder Theorem 3.1. It claims that instances of the teacher's problem are always solvable:

> A teacher has a number of pupils. Arranging them in rows of two shows that one pupil remains. With rows of five three remain. With rows of three again one pupil remains. Given that the class is of a standard size, how many pupils are there?

Mathematically we write that as follows. Denote by x the number of pupils. We know that

$$x \equiv_2 1, \quad x \equiv_5 3, \quad x \equiv_3 1.$$

The notation explains itself from the teacher's problem. (We read 'x is congruent to 1 modulo 2' and so on. Most people write $x \equiv 1 \bmod 2$ instead, but I don't like that.) Clearly, if x is a solution then also $x + 2 \cdot 3 \cdot 5 = x + 30$ is a solution and vice versa.

CHINESE REMAINDER THEOREM 3.1. *Suppose* $N = N_1 N_2$ *with* $\gcd(N_1, N_2) = 1$. *Then the (canonical) map*

$$\begin{array}{rcl} \mathbb{Z}_N & \longrightarrow & \mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}, \\ a \bmod N & \longmapsto & (a \bmod N_1, a \bmod N_2) \end{array}$$

is well defined, structure preserving, injective (1-1) and surjective. In other words: it is an isomorphism.

Here, we understand '$\operatorname{mod} N$' to be the canonical map $\mathbb{Z} \to \mathbb{Z}_N,\ a \mapsto a \operatorname{mod} N$ whatever implementation of $\mathbb{Z}_N$ you have chosen. This map respects 0, 1, addition and multiplication and it maps N to 0. Since the map in the theorem is not defined using $a \operatorname{mod} N$ for describing the image but a, we need to prove that everything is well defined. Namely, if $a \operatorname{mod} N = a' \operatorname{mod} N$ then the description of the image must not depend on whether we use a or a' to compute it, $(a \operatorname{mod} N_1, a \operatorname{mod} N_2) = (a' \operatorname{mod} N_1, a' \operatorname{mod} N_2)$.

A more classical formulation of the theorem states that for any $a_1, a_2 \in \mathbb{Z}$ there is some $x \in \mathbb{Z}$ such that $x \equiv_{N_1} a_1$ and $x \equiv_{N_2} a_2$. That is exactly the surjectivity in the above version which is actually the most tricky part. So let's try to prove that.

PROOF. It is easy to check that the map is well defined, structure preserving and injective. To see that it is surjective take some $a, b \in \mathbb{Z}$. We need to find some integer $x \in \mathbb{Z}$ such that $x \operatorname{mod} N$ maps to $(a \operatorname{mod} N_1, b \operatorname{mod} N_2)$. Since the map respects addition and scalar multiplication it is enough to consider the special cases $a = 1$, $b = 0$ and $a = 0$, $b = 1$. For if x_1 maps to $(1, 0)$ and x_2 maps to $(0, 1)$ then $x = ax_1 + bx_2$ maps to $a(1, 0) + b(0, 1) = (a \operatorname{mod} N_1, b \operatorname{mod} N_2)$ as desired. By symmetry we only consider the case $a = 1$, $b = 0$.

So our task is to solve $x = 1 - k_1 N_1$ and $x = 0 + k_2 N_2$ simultaneously for $x, k_1, k_2 \in \mathbb{Z}$. In particular, we need to solve

$$1 = k_1 N_1 + k_2 N_2. \tag{3.2}$$

Such an equation is the result of the extended euclidean algorithm applied to (N_1, N_2) provided their greatest common divisor is 1 as requested in the theorem, see Section 3.3. □

Actually, if the greatest common divisor of N_1 and N_2 is different from 1 then there is no solution to (3.2).

Further, we observe that

$$\begin{aligned} k_2 N_2 &\equiv_{N_1} 1, & \qquad k_1 N_1 &\equiv_{N_1} 0, \\ k_2 N_2 &\equiv_{N_2} 0, & \qquad k_1 N_1 &\equiv_{N_2} 1. \end{aligned}$$

Thus we can compute $x = ak_2 N_2 + bk_1 N_1$. This is called the Chinese Remainder Algorithm. Of course, we can generalize that to more than two moduli.

3.3. The extended Euclidean Algorithm. We do not consider the euclidean algorithm in detail. We simply perform an example. All calculations are displayed in a simple table. We are given two integers, say $a = 66\,013$, $b = 46\,199$. The aim is to compute their greatest common divisor g. At the same time we would like to represent it as a linear combination $g = sa + tb$. We have a pair (a, b) with a certain, still unknown greatest common divisor. We ask how we can change this pair such that the greatest common divisor of the new pair is still the same. First answer: $(b, a - b)$ can be the next pair. Better answer: $(b, a \operatorname{rem} b)$. Any common divisor of the old pair divides also $r_2 := a \operatorname{rem} b = a - q_1 b$. Any common divisor of the new pair divides also $a = q_1 b + r_2$. So we perform a division with remainder to make the pair 'smaller'. Finally, the greatest common divisor shall be represented as a linear combination of a and b. It is easy to represent a and b as such: $a = 1 \cdot a + 0 \cdot b$, $b = 0 \cdot a + 1 \cdot b$. And from that we easily get a representation of $r_2 = a - q_1 b$ as $s_2 = s_0 - q_1 s_1$, $t_2 = t_0 - q_1 t_1$. So we have this:

i	r_i	q_i	s_i	t_i	Comment
0	66 013	–	1	0	
1	46 199	1	0	1	
2	19 814		1	−1	$66\,013 \quad = 1 \cdot 46\,199 + 19\,814$

We have put $r_0 = a$ and $r_1 = b$. For each line we have $r_i = s_i a + t_i b$. Continuing this yields:

i	r_i	q_i	s_i	t_i	Comment
0	66 013	–	1	0	
1	46 199	1	0	1	
2	19 814	2	1	-1	$66\,013 = 1 \cdot 46\,199 + 19\,814$
3	6 571	3	-2	3	$46\,199 = 2 \cdot 19\,814 + 6\,571$
4	101	65	7	-10	$19\,814 = 3 \cdot 6\,571 + 101$
5	6	16	-457	653	$6\,571 = 65 \cdot 101 + 6$
6	5	1	7 319	$-10\,458$	$101 = 16 \cdot 6 + 5$
7	**1**	5	$\mathbf{-7\,776}$	**11 111**	$6 = 1 \cdot 5 + 1$
8	0	–	46 199	$-66\,013$	$5 = 5 \cdot 1 + 0$

Since the greatest common divisor of the last pair $(r_7, r_8) = (1, 0)$ is 1, this is the greatest common divisor of $(a, b) = (66\,013, 46\,199)$. In general, the last pair is $(g, 0)$ whose greatest common divisor is g. Since for each line r_i is represented by s_i and t_i we can read off:

(3.3) $$1 = -7\,776 \cdot 66\,013 + 11\,111 \cdot 46\,199.$$

The extra line provides a simple crosscheck: $0 = 46\,199 \cdot 66\,013 - 66\,013 \cdot 46\,199$ which is obviously true.

For the Chinese Remainder problem this gives us the two base solutions:

$$-7\,776 \cdot 66\,013 \equiv_{66\,013} 0, \qquad 11\,111 \cdot 46\,199 \equiv_{66\,013} 1,$$
$$-7\,776 \cdot 66\,013 \equiv_{46\,199} 1, \qquad 11\,111 \cdot 46\,199 \equiv_{46\,199} 0.$$

3.4. The unit group $\mathbb{Z}_N^\times$.

Looking at an earlier question we see that the extended euclidean algorithm also solves the question how to calculate inverses in $\mathbb{Z}_N$. Reading (3.3) modulo $66\,013$ we find:

$$46\,199 \cdot 11\,111 = 1 \quad \text{in } \mathbb{Z}_{66\,013}.$$

Thus $46\,199^{-1} = 11\,111$ in $\mathbb{Z}_{66\,013}$. Let us give a name to the set of invertible elements, also known as *units*, of $\mathbb{Z}_N$: We call it the *unit group* $\mathbb{Z}_N^\times$ of $\mathbb{Z}_N$. It always contains 1 but never 0. Since $1 + 1 + \cdots + 1 = 0$, the set of invertible elements is not closed under addition. But it is closed under multiplication! We can easily verify this by only using the axioms. Suppose $a, b \in \mathbb{Z}_N$ both have an inverse. Say, $ax = 1$ and $by = 1$. Then clearly, $(ab)(yx) = 1$. Thus ab has an inverse, namely yx. Further with a also its inverse x has an inverse, namely a. Thus $\mathbb{Z}_N^\times$ is, don't PANIC, a *commutative group*:

- It is **p**roperly defined: there is a set $\mathbb{Z}_N^\times$ with a multiplication $\mathbb{Z}_N^\times \times \mathbb{Z}_N^\times \to \mathbb{Z}_N^\times,\ (a, b) \mapsto ab$, a neutral element $1 \in \mathbb{Z}_N^\times$, and an inversion map $\mathbb{Z}_N^\times \to \mathbb{Z}_N^\times,\ a \mapsto a^{-1}$. (By abuse of notation, we use the same symbol for the set and for the group! In an object oriented programming language that would cause problems...)
- The multiplication is **a**ssociative.
- 1 is a **n**eutral element.
- Each element is **i**nvertible.
- The multiplication is **c**ommutative.

There are also groups that are not commutative. The smallest non-commutative group is the group of permutations on three points with only six elements. Rotations in space also form a non-commutative group, the operation is concatenation. Getting back to $\mathbb{Z}_N$, the extended euclidean algorithm allows us to compute an inverse whenever $x = (a \bmod N)$ and $\gcd(a, N) = 1$. Namely, then it yields an equation $1 = sa + tN$ and thus $1 = (s \bmod N)x$ in $\mathbb{Z}_N$. Vice versa, if $1 = yx$ and $y = b \bmod N$ then $1 = ba + tN$ for some $t \in \mathbb{Z}$. This implies that a and N have no non-trivial common divisors, since any common divisor of a and N divides $1 = ba + tN$. That is, $\gcd(a, N) = 1$. Concluding: the set $\mathbb{Z}_N^\times$ consists exactly of those elements that come from integers coprime to N; $\mathbb{Z}_N^\times = \{a \bmod N \mid a \in \mathbb{Z},\ \gcd(a, N) = 1\}$.

3.5. Repetition lengths in finite groups. Still, we have not found why $x^{1+L} = x$ in $\mathbb{Z}_N$ as we need for RSA. But by the Chinese Remainder Theorem 3.1 we only need to prove $x^{1+L} = x$ in $\mathbb{Z}_p$, and similarly in $\mathbb{Z}_q$. That might be easier since p and q are prime. And it actually is. For observe: For $x = 0$ the claimed equality is trivial. And all other elements in $\mathbb{Z}_p$ are invertible and thus in $\mathbb{Z}_p^\times$! (Indeed, given $0 \le a < p$, we have $a \bmod p$ invertible iff a is coprime to p, that is, $a \neq 0$.) That means, that we can work in a group instead of a domain where multiplication is only nice with respect to some but not all elements. That all elements but 0 have a multiplicative inverse is of course the same as saying that $\mathbb{Z}_p$ is a field. Recall that we announced to prove that at the beginning of Section 3.

We already observed that repeating a fixed operation, as multiplication by a fixed element x in $\mathbb{Z}_p^\times$, must lead to a finally periodic sequence $(1, x, x^2, x^3, \dots)$. Now, the number of elements in $\mathbb{Z}_p^\times$ is $p - 1$. Thus the repetition length is at most $p - 1$. Now, something miraculous happens: for any element x in the group $\mathbb{Z}_p^\times$ its size (also known as *order*) $\#\mathbb{Z}_p^\times = p - 1$ is a repetition length. In most generality this is

LAGRANGE'S THEOREM 3.4. *Suppose G is a (multiplicatively written) finite group and $x \in G$ any element. Then*

$$x^{\#G} = 1.$$

PROOF. We first give a proof only for commutative groups. Consider a list

$$g_1, g_2, g_3, \dots, g_{\#G}$$

of all group elements without any repetitions. Now, multiply all its elements by x and obtain the new list

$$xg_1, xg_2, xg_3, \dots, xg_{\#G}.$$

We claim that this also is a list of all group elements without repetitions.

First argument: no two elements on the new list are equal. Otherwise, $xg_a = xg_b$ for some indices $a \neq b$ and thus, being in a group, $g_a = g_b$ after multiplying by the inverse of x. Thus $a = b$, contradiction.

Second argument: the second list is complete. Take some group element g_a. Then $x^{-1}g_a = g_b$ for some index b since the first list is complete. But then $xg_b = g_a$ and so g_a is somewhere on the second list.

Since both lists are finite either of the preceding arguments would have been enough to prove that the lists are equal apart from their order. Now multiply all elements on each list. This yields

$$\begin{aligned} g_1 \cdot g_2 \cdot \dots \cdot g_{\#G} &= (xg_1) \cdot (xg_2) \cdot \dots \cdot (xg_{\#G}) \\ &= x^{\#G} g_1 \cdot g_2 \cdot \dots \cdot g_{\#G}. \end{aligned}$$

Multiplying with the inverse of the left hand side now gives $1 = x^{\#G}$. □

PROOF*. For the general result we must consider subgroups and cosets.

CLAIM. *Suppose H is a subgroup of G, that is, any subset of G closed under the group operations. Then $\#H$ divides $\#G$.*

First let us prove the claim. To this end we consider the *left cosets* $xH = \{xh \mid h \in H\}$.

Cosets are either equal or disjoint. Indeed, suppose $a \in xH \cap yH$. Then $a = xh' = yh''$ and thus $x = yh''(h')^{-1}$ and $xh = yh''(h')^{-1}h$ proving $xH \subset yH$. By symmetry then $xH = yH$.

All cosets are equal in size. Indeed, the map $H \to xH,\ h \mapsto xh$ is bijective since x has an inverse in G. Thus $\#(xH) = \#H$.

Any group element is in a coset. Indeed, $x \in xH$ since $1 \in H$ because H is a subgroup.

Thus G is a disjoint union of cosets, each of which has size $\#H$. Thus $\#H$ divides $\#G$. This proves the claim.

Now we finish the proof using the claim. Consider the subgroup of G generated by x, that is, $H = \langle x \rangle := \{\dots, x^{-2}, x^{-1}, 1, x, x^2, \dots\}$. Clearly, H is a group again. Further, the size of H is precisely the smallest repetition length L of x, also known as *order* $\mathrm{ord}(x)$ *of* x:

CLAIM. $\# \langle x \rangle = \mathrm{ord}(x)$.

Indeed, since $x^L = 1$ we may reduce exponents modulo L and we have $H \subset \{1, x, x^2, \dots, x^{L-1}\}$ and $\#H \le L$. And the smallest repetition length L of x can be at most $\#H$ of course. Thus they are equal.

By the first claim $\#H = L$ divides $\#G$, that is, $\#G = k \cdot L$ for some $k \in \mathbb{N}$. Now, $x^{k \cdot L} = (x^L)^k = 1^k = 1$ proving the theorem. □

Consider the unit group $\mathbb{Z}_N^\times$ again. Its size is denoted by $\varphi(N)$ and φ is called the *Euler totient function*. Applying Lagrange's Theorem 3.4 to this group yields

EULER'S THEOREM 3.5. *Suppose N is any positive integer and $x \in \mathbb{Z}_N^\times$. Then*

$$x^{\varphi(N)} = 1.$$

In other notation, if $a \in \mathbb{Z}$ is coprime to N we have $a^{\varphi(N)} \equiv_N 1$. □

Specializing to primes gives

FERMAT'S LITTLE THEOREM 3.6. *Suppose p is a prime and $x \in \mathbb{Z}_p^\times$. Then*

$$x^{p-1} = 1.$$

In other notation, if $x \in \mathbb{Z}$ is no multiple of p then we have $x^{p-1} \equiv_p 1$.

From this very last result, we now deduce the

COROLLARY 3.7. *Suppose p is prime and $k \in \mathbb{N}$. Then $x^{1+k(p-1)} = x$ for $x \in \mathbb{Z}_p$.*

PROOF. The claim is true for $x = 0$. Otherwise $x \in \mathbb{Z}_p^\times$ and thus by Fermat's Little Theorem 3.6 we have $x^{p-1} = 1$ and thus $x^{1+k(p-1)} = x(x^{p-1})^k = x \cdot 1 = x$. □

Now, finally we can give the desired nice proof of the RSA correctness:

THEOREM 3.8. *RSA is correct, that is, for any $x \in \mathbb{Z}_N$ we have $x^{ed} = x$.*

PROOF. Since $ed = 1$ in $\mathbb{Z}_L$ we have $\check{e}\check{d} = 1 + tL$ for some $t \in \mathbb{N}$. (Here, the conditions $e = \check{e} \bmod L$ and $0 \le \check{e} < L$ defines a structureless mapping $\check{~}\colon \mathbb{Z}_L \to \mathbb{Z}$.) By the Chinese Remainder Theorem 3.1 and symmetry we only need to prove $x^{ed} = x$ for $x \in \mathbb{Z}_p$. Since $L = (q-1)(p-1)$ using $k = t(q-1)$ in the previous result gives $x^{ed} = x^{1+t(q-1)(p-1)} = x$ in $\mathbb{Z}_p$. Together $x^{ed} = x$ in $\mathbb{Z}_N$. □

Generalizing this a little we can calculate the values of the Euler totient function from a factorization of its argument:

THEOREM 3.9. *Suppose $N = \prod_{i<t} p_i^{e_i}$ is given as a product of prime powers with different primes p_i. Then*

$$\varphi(N) = \prod_{i<t} p_i^{e_i-1}(p_i - 1) = N \prod_{\substack{p|N,\\ p \text{ prime}}} (1 - \frac{1}{p}).$$

PROOF. Based on the Chinese Remainder Theorem 3.1 and Fermat's Little Theorem 3.6 we can give a short proof:

In case $N = p$ is prime, the only non-invertible element of $\mathbb{Z}_p^\times$ is 0, all others have a trivial greatest common divisor with p and thus by the EEA an inverse modulo N. Consequently, $\varphi(p) = p - 1$ then.

In case N is a prime power, say $N = p^e$, inspection shows that $\varphi(p^e) = p^{e-1}(p - 1)$. (Do this as an exercise, we do not need that in this course.)

If $N = N_1 N_2$ with N_1 and N_2 coprime then the Chinese Remainder Theorem 3.1 tells us that $\mathbb{Z}_N$ is isomorphic to $\mathbb{Z}_{N_1} \times \mathbb{Z}_{N_2}$. In particular, invertible elements correspond to invertible elements. In other words, $\mathbb{Z}_N^\times \simeq \mathbb{Z}_{N_1}^\times \times \mathbb{Z}_{N_2}^\times$ and so their size is equal: $\varphi(N) = \varphi(N_1) \cdot \varphi(N_2)$.

By induction the previous statements prove the theorem. □

In the following we will encounter cryptographic systems that use some unspecified group generated by an element x. The system needs to know its order, its minimal repetition length. Actually, since the security might be based on it, we should at least have a way to check the order. Here it goes:

THEOREM 3.10. *Suppose G is a group, x is an element of G and L is some number. Then the minimal repetition length of x is L if and only if*

- ○ $x^L = 1$ *and*
- ○ *for each prime divisor p of L we have $x^{\frac{L}{p}} \neq 1$.*

PROOF. You can do this as an exercise. □

This turns into an order test provided you know all the prime divisors of L. (Computing them might be difficult!)

3.6. Discrete logarithms. The security of RSA is somehow based on the difficulty of factoring large integers. There is another very prominent problem from number theory that has similar properties. We already noted in Section 2.1.2 that the exponentiation map

$$\begin{array}{rcl} \mathbb{Z}_{\#G} & \longrightarrow & G, \\ e & \longmapsto & x^e \end{array}$$

is easy to evaluate via repeated squaring (aka. repeat and multiply). Its inverse map is called the *discrete logarithm*: $\operatorname{dlog}_x y = e$ means that $y = x^e$. The discrete logarithm problem in a group G is the problem to compute the discrete logarithm e from x and y. For many groups it is difficult to compute discrete logarithms.

For example, if p is a large prime such that we have a large prime factor q of $p - 1$ then the discrete logarithm problem in the subgroup G of $\mathbb{Z}_p^\times$ generated by some element of order q is probably difficult. On the one hand side, there is a particularly fast method for computing discrete logarithms in $\mathbb{Z}_p^\times$: the index calculus. (Index is another common name for a discrete logarithm.) The prime p must be large enough to prevent this. On the other hand side, there are several known generic algorithms. *Generic* means, laxly spoken, that the algorithm works in any group and does not use special features. The prime q must be large enough to protect against them. There is no (publicly) known way to use the special properties of $\mathbb{Z}_p$ for a small subgroup G.

Another example are elliptic curves. An elliptic curve is the set of solutions of a cubic equation, as for example $y^2 = x^3 + ax + b$ with given coefficients a and b, plus one point, called $\mathcal{O}$, at infinity. We can consider this kind over any field, not only over the reals or the complex numbers. For cryptography we choose a finite field $\mathbb{F}_q$. (To any prime power q there exists an essentially unique

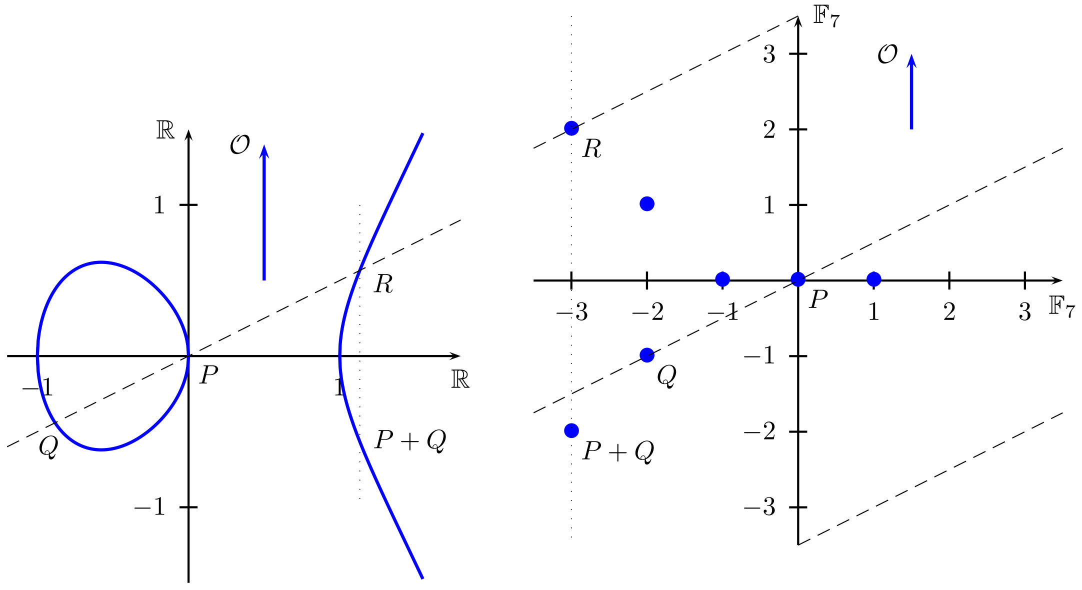

Figure 3.1: The elliptic curve $y^2 = x^3 - x$ over $\mathbb{R}$ and over $\mathbb{F}_7$

finite field with q elements. It is denoted by $\mathbb{F}_q$ or GF(q), for Galois field.) A lot of things are known about elliptic curves. The number of solutions to the cubic equation can be at most $2q$ since for each $x \in \mathbb{F}_q$ there are at most two possible values y. Actually, on average there is approximately one, the Hasse bound states that $|\#E - (q+1)| \leq 2\sqrt{q}$. The most miraculous property however is that these curves carry a group structure! To add two points P and Q simply 'draw' the line through them. Since the curve is the solution of a cubic equation there will be a third point R on this line. If you reflect this point at the x-axis you obtain a point $P + Q$, the sum of P and Q.[1] (Sorry for the inconvenience: Due to various reasons the operation on elliptic curves is called addition and not multiplication. So instead of multiplying we add, instead of taking the e-th power we compute the scalar multiple by e. Just remind yourself, that everything we do with a group is formulated using the group operation, regardless of its name.) There are some technical problems, for example, what happens when $P = Q$ or $P = -Q$, but they can all be solved. This turns the mere set into a group! And there is no known algorithm for computing discrete logarithms on most elliptic curves that is essentially faster than the generic algorithms. For that reason, the number of bits required for q is only between 160 and 240 rather than 1024 needed to store a group element in the previous example.

PS: There are groups where discrete logarithms are easy. In the additive group $\mathbb{Z}_p^+$ of the ring $\mathbb{Z}_p$ with a prime p, for example, computing a discrete logarithm is simply a multiplication with the inverse of the basis.

4. Signatures

Keywords: RSA signatures, ElGamal signatures, Schnorr signatures and refinements. Security.

Let us first ask what usual signatures are made for. A classical handwritten signature certifies that the signer accepts the statement he signs. For example, she agrees to some purchase contract or she states a final will. Later, anybody should — at least in principle — be able to verify whether the signature is valid. And the signature is thought to mean that actually the signer acknowledged the document's contents. Note that this can in some case turn against a signer. She can not deny that she signed and must comply to the statement made in the signed statement. In any case, the signature links a person, the signer, to the document. In practice, the signature is just the intermediary between person and document. Actually, the same will be true for electronic signatures. The signature a

[1]Consequently, $P + Q + R = \mathcal{O}$ is the neutral element.

priori only establishes a link between some document and some secret information. By other means, namely non-cryptographic ones, it must be guaranteed that the secret information is only accessible to a given person.

This last point is particularly problematic in case the person claims that she 'lost' this very secret information or that somebody has stolen it. But the non-electronic world already deals with this problem in connection with, say, credit cards or bank cards. The legal holder is simply made responsible for his secret information. In case of a theft or loss he has to announce this in time. Transactions or in our case signatures made in the mean time are (at least partially) attributed to the holder. By this the legal holder is forced to be careful and cannot simply deny that he gave a certain signature. Clearly, there is no cryptographic solution to this problem.

The cryptographic problem that we are facing is how to link secret information to a given — of course electronic — document.

Section 2.5.1 in Kou (2003), page 25ff, gives a short overview over signatures:

- ○ RSA signature scheme (Use decoding to sign),
- ○ Rabin public key signature scheme (square roots mod $p \cdot q$; complete break equivalent to factoring; but *completely broken* by chosen-message attack),
- ○ DSA (Schnorr signatures),
- ○ ECDSA,
- ○ blind digital signatures,
- ○* undeniable signatures,
- ○* fail-stop signatures,
- ○* group signatures,
- ○* proxy signatures.

4.1. RSA signatures. In RSA Alice chooses a key pair (N, e) and (N, d) such that $x^{ed} = x$ for every $x \in \mathbb{Z}_N$. Anybody can compute the encryption $y = x^e$ of some (number encoding a) message $x \in \mathbb{Z}_N$. But only Alice can compute the decryption of some number $y \in \mathbb{Z}_N$. Moreover, the order of encryption and decryption does not matter. Thus the signature could work as follows.

- ○ Decrypt the document $x \in \mathbb{Z}_N$. Now (x, x^d) is the signed document. (Actually, the x is superfluous since it can be computed from x^d.)

- ○ To verify a signed document $(x, y) \in \mathbb{Z}_N^2$ simply use the encryption and test whether $x = y^e$.

Actually, we can easily produce a document that is seemingly signed by Alice: (y^e, y). However, we only have very little control on the document $x = y^e$ then. This seems not to be a really great threat but still: If Mallory chooses some number (more or less at random) $r \in \mathbb{Z}_N$ and makes Alice sign $r^e \cdot x$, he obtains $y = r^{ed}x^d$. From that he can easily extract x^d since $r^{ed} = r$ and thus he has a signature of a document Alice has never seen.

Further problems occur in connection with signing encrypted documents.

A countermeasure is to introduce a hash function $h\colon \{0,1\}^* \to \mathbb{Z}_N$. Instead of signing x itself we sign its hash value $h(x)$. This also makes the signature typically much smaller since most hash values use only 160 to 320 bits regardless of the size of the document:

- ○ In order to sign x Alice decrypts the document's hash value $h(x) \in \mathbb{Z}_N$. Now $(x, h(x)^d)$ is the signed document.

- ○ To verify a signed document $(x, y) \in \mathbb{Z}_N^2$ simply use the encryption and test whether $h(x) = y^e$.

Now, a forger must solve the equation

$$h(x) = y^e \quad \text{in } \mathbb{Z}_N \tag{4.1}$$

where he may use a signature for a different document x'.

One way of forging breaks the hash function and computes x for a given y. To prevent forging a signature we must now require that it is difficult to find x with a given hash value y^e. Actually, it must even be impossible to find two documents $x \neq x'$ with the same hash value $h(x) = h(x')$. If we are able to do the latter and maybe large parts of x and x' even coincide, then we could let Alice sign the innocuous document x' obtaining $y = h(x)^d$ and later we present (x, y) where Alice, now president, admits that she forged the latest president elections.

Another way of forging of course still consists in breaking RSA. Yet, we assume that this is difficult. It does not really help us to use a random factor to blur the message, since (hopefully) the hash function is not multiplicative.

There may be other ways to solve (4.1). It would be nice if we could prove that any way of doing so involves either breaking the hash function h or the RSA encryption. But no such proof is known or even suspected.

4.2. *Rabin signatures. Rabin signatures are very similar to RSA signatures. But they use $e = 2$ and thus a corresponding d does not exist. Sometimes however there are ways to extract square roots modulo some number N. For example, if $N = p \cdot q$ and both p and q are congruent to 3 modulo 4. Then $y = x^{\frac{p+1}{4}}$ is a square root of $x \in \mathbb{Z}_p$ if one exists. [The square of y is $x^{\frac{p+1}{2}}$. Now suppose that $x = z^2$. Then $y^2 = z^{p+1} = z^2 = x$. Actually, we have either $y = z$ or $y = -z$.] Thus by Chinese Remainder Theorem 3.1 you can extract square roots modulo N provided you do know p and q.

This system is nice since one can prove that forging a signature is equally difficult as computing the factorization of N. But at the same time this feature *completely breaks* the system by the following chosen message attack: Suppose Mallory chooses some message x^2 and obtains a signature s. Then $x^2 = s^2$. By the Chinese Remainder Theorem 3.1 there are four square roots of x^2. And there is a fifty percent chance that neither $x = s$ nor $x = -s$. Then Mallory knows $N|(x-s)(x+s)$ but N divides none of the factors. Thus gcd$(N, x - s)$ must be a proper factor of N.

EXERCISE 4.2 (Chosen-message attacks for Rabin signatures). *The evil grand vizier Jaffar wants to find out the signature of the enchanting princess Jasmin! She uses Rabin's signature scheme and gladly gives autographs to honest-looking strangers. Naturally Jaffar is only satisfied if he finds out the prime factors p and q of Jasmin's number N. Jaffar chooses a random number x, computes x^2 and asks Jasmin to sign the message x^2 for him.*

(i) Suppose we have $x, s \in \mathbb{Z}$ with $x^2 \equiv_N s^2$ and $x \not\equiv_N \pm s$. Show that N divides $x^2 - s^2$ but neither $x - s$ nor $x + s$.

(ii) Deduce that gcd$(N, x - s)$ *is a proper factor of N, that is, a factor that is neither 1 nor N.*

(iii) Compute the probability that he can find out the prime factors p and q this way.

Hint*: Chinese Remainder Theorem.*

(iv) How large is the probability that Jaffar finds out the prime factors p and q after k iterations of this strategy?

(v) Deduce:

THEOREM. *Rabin's signature scheme is completely broken by a chosen-message attack.*

Alas, Genie will have to come to the rescue of princess Jasmin. Or maybe Aladdin can help . . .

4.3. General ElGamal signatures. The much too nice properties of RSA ask for different signature schemes that do not allow for splitting off random factors as above. The following system gives a different signature scheme.

- We fix some (cyclic) group G and a generator g where the discrete logarithm problem is difficult. Let L be the order of g. Further fix an arbitrary map $G \to \mathbb{Z}_L,\ a \mapsto \widetilde{a}$. (For example, read the bit representation of the group element as the bit representation of an integer and reduce that modulo L.)

- ◦ Alice chooses a secret key $x \in \mathbb{Z}_L$ and computes $y = g^x \in G$ as a public key.
- ◦ Sign an element $m \in \mathbb{Z}_L$, say an encoded message or a (cryptographically secure) hash value of such:

  Choose $k \in \mathbb{Z}_L^\times$ and compute $a = g^k \in G$. Now it is easy for Alice to find $b \in \mathbb{Z}_L$ such that

  $$y^{\widetilde{a}} a^b = g^m \in G, \tag{4.3}$$

  since this is equivalent to the equation $x\widetilde{a} + kb = m \in \mathbb{Z}_L$ and the only real task is to invert k in $\mathbb{Z}_L$. Now, (a, b) is the signature for m.
- ◦ Verify a signed document (m, a, b):

  Verify the equation (4.3). Note that all necessary information to do that is publicly available!

Correctness: clear. Efficiency: fast. Security: Solving (4.3) is probably difficult. All proposed algorithms require finding some discrete logarithm. The size of the signature is quite large: An element in G and an element in $\mathbb{Z}_L$ need almost twice as much space as the signed element m. If you choose G as a subgroup of 160-bit prime order of $\mathbb{Z}_p^\times$ with $1\,024$-bit prime p then the signature needs $1\,184$ bits for a 160-bit element.

4.4. General Schnorr signatures. Schnorr invented a variation of the ElGamal signatures that produces even smaller signatures.

PROTOCOL 4.4. General Schnorr signature scheme.

1. Fix a prime p so large that discrete logarithms are difficult in $\mathbb{Z}_p$ and a prime factor q of $p - 1$ so large that a birthday attack (Pollard rho) is infeasible. Choose an element g of order q. (Choose $h \in \mathbb{Z}_p^\times$ arbitrary, let g be $h^{\frac{p-1}{q}}$. If this turns out to be 1 then retry.) Let $Q\colon \mathbb{Z}_p \to \mathbb{Z}_q$ map an element of $\mathbb{Z}_p$ to its smallest non-negative representative reduced modulo q. (This map has no nice structure, it is neither additive nor multiplicative.)
   More generally, fix a group H (instead of $\mathbb{Z}_p^\times$) and a generator g of prime order q. The discrete logarithm problem in the group H must be difficult, in particular the order q must be large enough to prevent a birthday attack (Pollard rho). Note that now $\mathbb{Z}_q$ is the exponent group of $G = \langle g \rangle$. Further fix a map $G \to \mathbb{Z}_q,\ h \mapsto Q(h)$.
2. Alice chooses a secret key $x \in \mathbb{Z}_q$ and computes $y = g^x \in G$ as a public key.

Sign an element $m \in \mathbb{Z}_q$:

3. Choose $k \in \mathbb{Z}_q$, let r be $Q(g^k)$ and solve

   $$Q(g^{ms^{-1}} y^{rs^{-1}}) = r. \tag{4.5}$$

   This is possible since (4.5) is implied by $m + xr = ks$ in $\mathbb{Z}_q$. (If $Q\colon G \to \mathbb{Z}_q$ is injective [one-to-one], the two equations are even equivalent.) Now, (r, s) is the signature for m.

Verify a signed document (m, r, s):

4. Verify the equation (4.5). Note that all necessary information to do that is publicly available!

For the digital signature scheme (DSS), $H = \mathbb{Z}_p$, p is a 512- to $1\,024$-bit prime, q is a 160-bit prime and Q is simply taking the smallest non-negative representative modulo q. Further, the digital signature algorithm (DSA) always signs the secure hash algorithm (SHA) hash value $m = \mathrm{SHA1}(\text{message})$ of the message.

To prevent some weak moduli the primes for DSS must be generated by a given algorithm. There is an algorithm, involving SHA again, that takes a random 160-bit sequence S and outputs q, a certain counter C and p satisfying the requirements above or it fails. Repeat it until it succeeds and retain

S and C as a prove that the primes were generated as required. (See Schneier (1996), section 20.1, page 489, for a complete description.)

Correctness: clear. Efficiency: fast. Space: 320 bits for a 160 bit hash value. Security: Solving (4.5) is probably difficult. All obvious algorithms require solving some discrete logarithm problem in H.

EXAMPLE 4.6. For DSA first you have to fix a prime q (160 bit or so), say $q = 11$. Then you need another prime p such that $q|(p-1)$ (1024 bit or so), say $p = 67$. Further we need an element of order 11 in $\mathbb{Z}_{67}^{\times}$: take some random element, say $h = 2$ then $g = 2^{66/11} = 2^6 = 64 = -3$ is an element of order (minimal repetition length) 11. Finally, we need some map Q which maps powers of g to elements of $\mathbb{Z}_{11}$, this will be just taking the minimal non-negative representative in $\mathbb{N}_{<67}$ and reducing it modulo 11. For example $Q(-3 \bmod 67) = 64 \bmod 11 = -2 \bmod 11$. This is the general setup, so we have

$$p = 67, q = 11, g = -3.$$

Then Alice needs a key pair, say she chooses $x = 4$, then $y = g^4 = (-3)^4 = 14$.

To sign the message $m = 3$ Alice chooses at random $k = 5$ and computes $r = Q(g^k) = Q((-3)^5) = Q(-42) = Q(25) = 25 \bmod 11 = 3$. Now she must solve the key equation

$$Q(g^{ms^{-1}} y^{rs^{-1}}) = r$$

for s. To do so she solves $m + xr = ks$ in $\mathbb{Z}_{11}$, that is, $3 + 4 \cdot 3 = 5 \cdot s$. The result is $s = 3$. Thus the signed document is $(m, r, s) = (3, 3, 3)$.

Bob verifies the key equation: $s^{-1} = 4$ (in $\mathbb{Z}_{11}$!), $ms^{-1} = 1$, $rs^{-1} = 1$, $g^1 y^1 = (-3) \cdot 14 = -42 = 25$, $Q(25) = 3$, and this is r so everything is fine. $\Diamond$

4.5. Attacks against the randomization (k). Each signature requires a new value of k. If Eve ever recovers a k that Alice used for signing, perhaps by exploiting some properties of the pseudo random number generator, then she can compute Alice' private key x. Actually this secret key is only protected by the secrecy of the second unknown in the equation $m + xr = ks$ which implies (4.5). If Eve ever gets two messages signed using the same k then she can recover x. [The two equations $m_1 + xr = ks_1$ and $m_2 + xr = ks_2$ are a 2×2 linear equation system over $\mathbb{Z}_q$ with respect to the indeterminates x and k.]

4.6. *Subliminal channels. A signature is not supposed to carry any additional information. It only links the given document to a certain secret. But there are ways of abusing signatures to send information secretly. For example, a malicious agent may distribute a program environment that allows you to sign documents with your secret key but at the same time with each signature leaks some bits of your secret key. Actually, you will not be able to detect that in the signatures, even if you suspect such a fraud. The only way to prove this attack would be to disassemble the program's code (provided it has not deleted its bad parts after completing its perfidious task). An agent might also use this channel to send secret messages when nobody thinks he is distributing information at all. This kind of trick is called a subliminal channel.

Schneier (1996) lists several:

- Any signature scheme with a random value allows to embed a few bits in the signature by choosing the random value until the signature has prescribed bits with the wanted values.
- ElGamal: Alice to anybody knowing her secret key.
- DSA: Alice to anybody knowing her secret key.
- DSA: A single bit. Or some bits.

Using the secret k the signer can embed secret messages in her signatures. Say, she has fixed some additional prime t different from p and q. With DSA, for example, she can choose k such that r is a quadratic residue modulo t or not (that is, $r \equiv_t s^2$ for some $s \in \mathbb{N}_{<t}$ or for no $s \in \mathbb{N}_{<t}$, respectively) depending on whether she wants to transmit a 1 or a 0. By using several different such extra primes even several bits (say up to ten or so) can be embedded. Similarly, Alice' signature

smart card could choose k randomly until some bits of r equal some given values. The embedded bits could be one-time pad encoded bits of her secret key. Only the card issuer would know of this and could extract Alice' key bits, it would not even be possible to detect this leak. Since the card can precalculate k this would not even be noticeable by longer signing times.

If Alice conspires with Bob she can even transfer as many bits as the signature through such a channel. With ElGamal, for example, she shares her secret key with Bob and simply uses the secret value k as the subliminal message. Knowing the secret key Bob can recover the message. Nobody else will even doubt that a message was sent to Bob.

All these subliminal channels can be foiled by restricting the choice of the random parameters.

5. Solutions for electronic money

Chaum. Brands. Further systems for additional properties like divisibility or smart card 'agencies'.

5.1. Chaum's system. See Schneier (1996), section 6.4, pages 139ff.

PROTOCOL 5.1. Chaum's electronic cash.

1. Alice prepares 100 anonymous money orders for 1 000 € each with a uniqueness string (that is, a serial number). On each order she adds a list of 73 pairs of identity bit strings, so that xoring a pair gives Alice' identity information, her name and account number, say. Alice commits to each of these 146 bit strings. Alice blinds each order and hands them to the bank.
2. The bank asks Alice to open all but one including all the identity string pairs and checks whether all data are as required. If so it signs the remaining order blindly.
3. Now Alice has a valid coin.

4. For spending it to the merchant Martin, she hands him the coin.
5. Martin verifies the bank's signature. If it is wrong he refuses to accept and calls the Police.
6. Then he chooses 73 random bits and asks Alice to open the left or right half of the identity pairs accordingly.
7. Alice does so.

8. Martin takes the money to the bank.
9. The bank verifies all constraints:
   - its signature,
   - the uniqueness string,
   - the identity strings.

   If the signature is wrong the bank refuses and calls the Police.
   If the uniqueness string is registered and the identity strings are opened as in the earlier coin, Martin tries to cheat.
   If the uniqueness string is registered and the identity strings are differently opened, the bank reconstructs Alice' identity information and calls the Police.

This system is *anonymous*: the bank cannot identify Alice by the information she gets from the merchant. And it is *offline*: It does not require an interaction with the bank during the payment. There is a protection against *double spending*: If Alice spends a coin twice she is caught by the identifier. If the bank detects a double spending it can distinguish whether Martin or Alice tried to cheat: If the bank receives a coin with the same uniqueness string but different identity strings then Alice tried to cheat. If also the identity strings are equal then Martin has to be blamed. Also Alice cannot frame Martin since she cannot control how Martin chooses his challenge. Only an alliance of Alice and Mallory, another merchant, may achieve this: if Mallory simply asks the same challenge as Martin and he is faster to be at the bank then ... To prevent this last kind of fraud the merchant can be forced to use special challenges that depend on his account number and the present date and time.

5.2. Chaum, Fiat & Naor (1989). Chaum's original cut-and-choose variant has the major disadvantage that the probability of cheating by using a faked envelope is still $\frac{1}{n}$ if n envelopes are used. Further, the last section does not yet tell us how to hide the identity information in a practical implementation such that Alice is really bound to it.

The following set of protocols gives an answer to these questions. We assume that the bank has published its public RSA key $(N, 3)$ and a security parameter k (specifying the number of 'envelopes' and 'identity splits' to use). For simplicity, let k be a multiple of 4. And the bank has fixed two collision-resistant functions $f, g\colon \mathbb{Z}_N \times \mathbb{Z}_N \to \mathbb{Z}_N$ and such that g with any fixed first argument gives a one-to-one (bijective) map. Further, Alice has opened the account number u and obtained a counter v that has to be advanced in every withdrawal.

PROTOCOL 5.2.

1. Alice chooses $a_i, b_i, c_i, d_i, r_i \in_R \mathbb{Z}_N$ for $i \in \mathbb{N}_{<k}$ at random under condition that

   $$a_i \oplus b_i = u||(v+i). \tag{5.3}$$

   Here $x \oplus y$ means the binary XOR of the binary representations of the smallest non-negative integers that reduce to x or y. (Write $x = \left(\sum_{j\in\mathbb{N}_{<\lceil \log_2 N\rceil}} x_j 2^j\right) \bmod N$ with $x_j \in \{0,1\}$ such that the integer $\sum_{j\in\mathbb{N}_{<\lceil \log_2 N\rceil}} x_j 2^j$ is less than N. Then use $x_j \oplus y_j$ to define $x \oplus y = \sum_{j\in\mathbb{N}_{<\lceil \log_2 N\rceil}} (x_j \oplus y_j) 2^j$. To avoid difficulties with the allowed range we might require that the topmost bit (the highest significant bit of N) of a_i, b_i and u is always zero.) With $u||(v+i)$ we mean the number with the binary representation $u \cdot 2^{32} + (v+i)$ supposing that we need 32 bits for the counter. Alice computes

   $$\begin{aligned} x_i &:= g(a_i, c_i), \\ y_i &:= g(b_i, d_i), \\ B_i &:= r_i^3 f(x_i, y_i) \end{aligned} \tag{5.4}$$

   for each i and sends the envelope vector (B_i) to the bank. $\xrightarrow{(B_i)_{i\in\mathbb{N}_{<k}}}$
2. Now the bank chooses a random subset R of $k/2$ indices in $\mathbb{N}_{<k}$ and sends R to Alice. $\xleftarrow{R}$
3. Alice opens the envelopes chosen by R by sending $(a_i, b_i, c_i, d_i, r_i)_{i\in R}$ to the bank. $\xrightarrow{(a_i,\dots)_{i\in R}}$
4. The bank tests (5.3) and (5.4) for $i \in R$. If this turns out well she computes

   $$s := \prod_{i\notin R} B_i^{(1/3)}$$

   and sends s to Alice. The bank charges Alice's account 100 € and increments the counter v by k. $\xleftarrow{s}$
5. Alice can then easily unblind this signature and obtains $C = s/\prod_{i\notin R} r_i$. She re-indexes the identity pairs (a_i, b_i, c_i, d_i) to the indices $0, \dots, k/2-1$ such that $\check{f}(x_0, y_0) < \check{f}(x_1, y_1) < \dots < \check{f}(x_{k/2-1}, y_{k/2-1})$. Finally, Alice increments her copy of v by k. Now she has a coin

   $$\Big((a_i, b_i, c_i, d_i)_{i\in\mathbb{N}_{<k/2}}, C\Big)$$

   which fulfills the condition that

   $$\begin{aligned} a_i \oplus b_i &= u||V(i), \\ x_i &= g(a_i, c_i), \\ y_i &= g(b_i, d_i), \\ C^3 &= \prod_{i\in\mathbb{N}_{<k/2}} f(x_i, y_i). \end{aligned} \tag{5.5}$$

To withdraw a coin with invalid identity information Alice would have to send some wrong B_i to the bank. But if she does so in only ε of all her envelopes then the chance of not being caught is $\binom{k(1-\varepsilon)}{k/2}/\binom{k}{k/2}$, which is at most $(1-\varepsilon)^{k/2} \le \mathrm{e}^{-\frac{1}{2}\varepsilon k}$. On the other hand to hide her identity in double spending she must get a pair of challenges that differs only on that ε proportion, thus the chance of getting a challenge that allows her to re-spend a coin is at most $\left(\frac{1}{2}\right)^{(1/2-\varepsilon)k}$. With $k = 128$ we get the following probabilities:

ε	Chance of forgery	Chance for successful double spending
0	1	2^{-64}
$1/8$	$2^{-17.54\ldots}$	2^{-48}
$1/4$	$2^{-39.55\ldots}$	2^{-32}
$1/2$	$2^{-124.17\ldots}$	1

To forge a coin without cheating the bank Alice would have to produce such a set of information fulfilling (5.5). If Alice can forge bank signatures that is no problem, she just computes a valid C. Otherwise she must adapt the right hand side to give a cube with a known root. Say, as a particular case, she chooses all but a_0, b_0, c_0, d_0 in advance. Then the remaining task is to find these to give a particular value for $f(g(a_0, c_0), g(b_0, d_0))$. But that means that Alice has an efficient way to compute preimages of that combination of f and g and therefore of f. That would mean that f is not one-way and thus not collision-resistant.

A third possibility would be to withdraw a valid coin but later use different values for the coin values that do not reveal the true identity but some garbage. Alice could do that if she knew collisions for g. Suppose $g(x, y) = g(x', y')$ with $x \neq x'$ is one such collision. Then she uses $a_0 = x$, $c_0 = y$ with the bank but when she spends the coin she uses $a_0 = x'$, $c_0 = y'$. Yet, finding collisions for g is supposed to be difficult. Of course, Alice has more room for variations but no variation seems to help her circumvent breaking g, f or RSA. Yet, we cannot prove that rigorously!

Now, let us see how to pay Martin:

PROTOCOL 5.6.

1. Alice sends the coin signature C to Martin. $\xrightarrow{C}$
2. Martin chooses some random bit string $z \in \{0,1\}^{k/2}$ and sends it to Alice. $\xleftarrow{z}$
3. Alice computes her answer Z by revealing a half of each identity pair:

$$Z_i = \begin{cases} (a_i, c_i, y_i) & \text{if } z_i = 1, \\ (x_i, b_i, d_i) & \text{if } z_i = 0. \end{cases}$$

   She sends Z to Martin. $\xrightarrow{Z}$
4. Martin computes $x_i = g(a_i, c_i)$ or $y_i = g(b_i, d_i)$ according to the value of z_i. He then checks the signature $C^3 = \prod_{i \in \mathbb{N}_{<k/2}} f(x_i, y_i)$ according to (5.5). If everything is OK, he accepts the payment.

The protocol already guarantees that the equations (5.5) must be valid. Otherwise Martin does not accept the coin. Of course, Martin wants to deposit the coin at the bank which is simply done as follows:

PROTOCOL 5.7.

1. Martin sends the entire payment transcript (C, z, Z) to the bank.
2. The bank verifies that the coin is valid and then checks whether the coin has already been deposited by searching for a coin with the same C in her database. If she does not find the coin she puts 100 € on Martin's account and sends him a receipt.
   If, however, she finds a coin (C, z', Z') she detects a double spending. There are two cases:
   - If $z = z'$ and $Z = Z'$ then Martin tries to redeposit an already deposited coin.
   - If $z \neq z'$ then also $Z \neq Z'$ and the bank knows a complete quadruple (a_i, b_i, c_i, d_i) and $a_i \oplus b_i = u||v'$ reveals Alice' identity. The bank calls the police.

   The case $z = z'$ and $Z \neq Z'$ is highly improbable. If transmission errors can be excluded this can only happen if Alice knows a collision for g.

There are various possible scenarios of trying to cheat. One possible problem is that Alice cooperates with Mallory, another merchant. She tells him to use Martin's challenge and then Mallory goes to the bank with the very same transcript as Martin. The bank knows that either Mallory or Martin is lying but she cannot tell which one. And also she has no way to catch Alice. But this is no new story to us: it can be prevented by using a pseudo-random challenge z that depends on the merchant's account and date and time of the transaction. Then Mallory cannot simply use the same challenge for he would be easily spotted as the misbehaving merchant and sued.

One further problem is that so far the bank can easily frame Alice for double spending. She can simply perform all of Alice' and Martin's actions. This means that the scheme cannot have any legal significance and thus no bank will use the system as it was presented. To prevent that Alice simply signs her identity in u. Instead of (5.3) Alice uses

$$a_i \oplus c_i = u || \operatorname{sig}_A(u, s_i) || (v + i).$$

Note that Alice must use a new random value s_i for each signature to prevent the bank from simply copying her signed identity.

5.3. Brands (1999). The solution described in Brands (1999) is a system that uses a different supposedly hard problem as the basis of an e€ system. Brands has described several variants of this system with different focuses. Apart from a system with similar properties than the previous one there are also solutions that incorporate a smart card as an extended arm of the bank. The smart card has to be asked upon any payment and can prevent double spending a priori. But even if the smart card's secret is revealed to a malicious Alice she still has to face double spending detection as in the previous system.

The basis for Brands' system is a generalization of the so-called discrete logarithm problem. If you work in a group then exponentiation is easy to calculate. However, finding an exponent e satisfying $x^e = y$ in the group may be difficult. An example for groups with supposedly difficult discrete logarithm problem are the subgroups of $\mathbb{Z}_p^\times$ generated by an element of order q where both p and q are prime and large enough. For example, taking p as a 1024-bit prime and q as a 160-bit prime was considered to be safe a few years ago. Other groups with difficult discrete logarithm problem are elliptic curves of appropriate size. (The number of bits for a point should be 160 to 240 bits.) The generalization used here is called the *representation problem*.

Suppose you are given several generators $g_1, \ldots, g_r$ and some $x \in G$.

Find $e_1, \ldots, e_r \in \mathbb{N}_{<\#G}$ with

$$x = g_1^{e_1} g_2^{e_2} \ldots g_r^{e_r}.$$

In case $r = 1$ this is simply the discrete logarithm problem, so we only use $r \geq 2$ here. Clearly, if we can solve the discrete logarithm problem in G then we can solve this problem. The inverse is only partially true. So assuming that this problem is difficult is a little more than assuming that finding discrete logarithms is difficult.

If you want to know more about the details then read section 4 in Brands (1999). The basic reasonings about how to detect double spending or to prevent framing Alice or … are similar than the one before.

In view of the next system let us emphasize one more point. This system does not use cut-and-choose to give the bank the necessary conviction that the final coin has Alice' identity embedded. Instead a clever use of exponents and generators guarantees that.

5.4. Ferguson (1994b). This system is based on the difficulty of RSA and a discrete logarithm problem but it also uses some hash functions at sensitive places to (hopefully) increase the security. Further, polynomial secret sharing is used in order to decrease the coin size without loss of security. The important part here is Martin's challenge size, it must be large enough to prevent repetitions. The challenge size in Chaum *et al.* (1989) was $k/2$ bits, so the size of the coin grows linearly with the wanted challenge size. Here the challenge size depends only on the chosen group and is thus typically not much larger than with, say, $k = 4$. But let us first explain the polynomial secret sharing and the system.

5.4.1. Polynomial secret sharing. Suppose there is some secret x that we want to give to a group of people. Yet, the secret is very valuable and we do not trust a single person far enough to give him the secret. Think of the access code of the central safe of a bank or the start code of nuclear weapons. The solution is to distribute the secret: each person only gets part of the secret. Now, we know that to determine a polynomial f of degree less than k over some field $\mathbb{F}$ we need to know k pairs $(x, f(x))$. By interpolation we can then recover f, in particular, say, $f(0)$. If we give one point $(x, f(x))$, $x \neq 0$, to each person then at least k of them must come together to recover the secret $f(0)$ and thus to be able to open the safe or to start the missile. Figure 5.1 shows a picture of a line over $\mathbb{F}_{257}$. Any two points determine the secret. But if we only know one point then any secret could complete the picture. In Figure 5.2 we see a line over $\mathbb{F}_{256}$, the elements of $\mathbb{F}_{256}$ have been numbered in some systematical way for that purpose. Again any two points determine the line, one point could go with any secret. Figure 5.3 shows cubic curves. Only if we know at least four of its non-zero points then we can recover the secret.

5.4.2. The system. Following the description of the author we also first describe the payment thus specifying the form of the coins. For the payment process we then have to find a way of getting the appropriate blind signatures from the bank. The basic setup contains an RSA signature key pair of the bank with public key (N, v). Additionally to the standard assumptions we require that v is a sufficiently large prime and that $\varphi(N)$ contains at least one large prime factor. Further some elements $g_1, g_2, g_3 \in \mathbb{Z}_N^\times$ of large order (minimal repetition length) are fixed. To be able to find them the bank should construct her primes p, q such that she knows large prime factors of $p-1$ and $q-1$. Next we need a suitable prime t such that $N \mid t-1$ and elements $h_2, h_3 \in \mathbb{F}_t^\times$ of order N. Finally, the bank chooses hash functions $f_1 \colon \mathbb{Z}_N^\times \to \mathbb{N}_{<v}$, $f_2, f_3 \colon \mathbb{F}_t^\times \to \mathbb{N}_{<v}$, and $f_4 \colon \mathbb{N}_{<v} \times \mathbb{N}_{<v} \to \mathbb{Z}_N^\times$. The bank publishes the data

$$(N, v, g_1, g_2, g_3, t, h_2, h_3, f_1, f_2, f_3, f_4).$$

Further Alice' identity is coded in a value $U \in \mathbb{N}_{<v}$. Note that we will do a lot of calculations in the RSA domain $\mathbb{Z}_N^\times$ but some calculations also will take place in the field $\mathbb{F}_t$.

The coin consists of randomly chosen values $a, b, c \in \mathbb{Z}_N^\times$ from which anybody can compute $A = a g_1^{f_1(a)}$, $B = b g_2^{f_2(h_2^b)}$, $C = c g_3^{f_3(h_3^c)}$. Further a random parameter $k \in \mathbb{N}_{<v}$ and signatures $S_1 = (AC^k)^{(1/v)}$ and $S_2 = (BC^U)^{(1/v)}$ are part of the coin.

PROTOCOL 5.8. Payment.

1. Alice hands over (a, b, c) to Martin. $\xrightarrow{(a,b,c)}$
2. Martin chooses a random challenge $x \in \mathbb{N}_{<v}$. $\xleftarrow{x}$
3. Alice computes $r + \widehat{r}v \leftarrow kx + U$ with $r \in \mathbb{N}_{<v}$ and a signature R to $A^x BC^r$ by $R \leftarrow S_1^x S_2 C^{-\widehat{r}} = (A^x BC^r)^{(1/v)}$. She sends (r, R) to Martin. $\xrightarrow{(r,R)}$
4. Martin verifies that the signature is valid: all transmitted data are in the required domains and
$$R^v \stackrel{?}{=} A^x BC^r.$$
Note that he can do that.

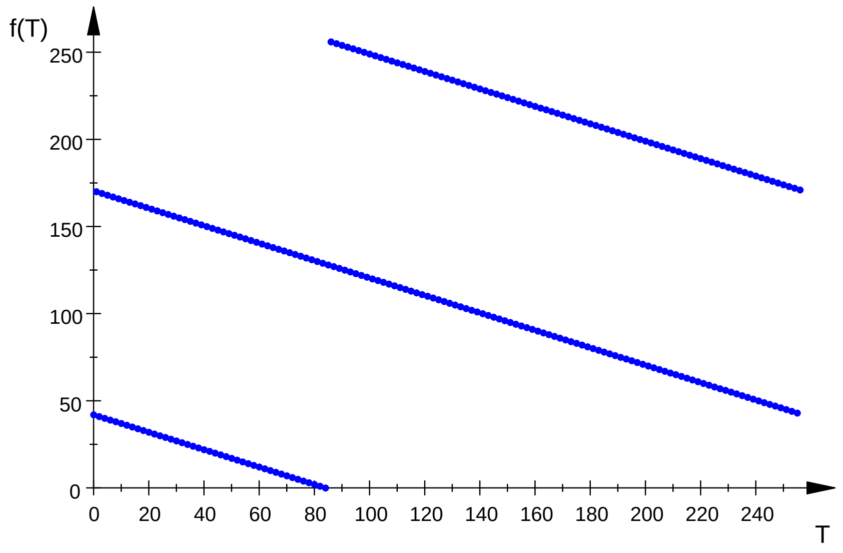

Figure 5.1: The line $f\colon \mathbb{F}_{257} \longrightarrow \mathbb{F}_{257},\ T \mapsto 128\,T + 42$ over the field $\mathbb{F}_{257}$ carries the secret $f(0) \mathrel{\widehat{=}} 42$ and passes through zero at $T \mathrel{\widehat{=}} 84$. The elements of $\mathbb{F}_{257}$ are represented as integers modulo 257 (which is prime!).

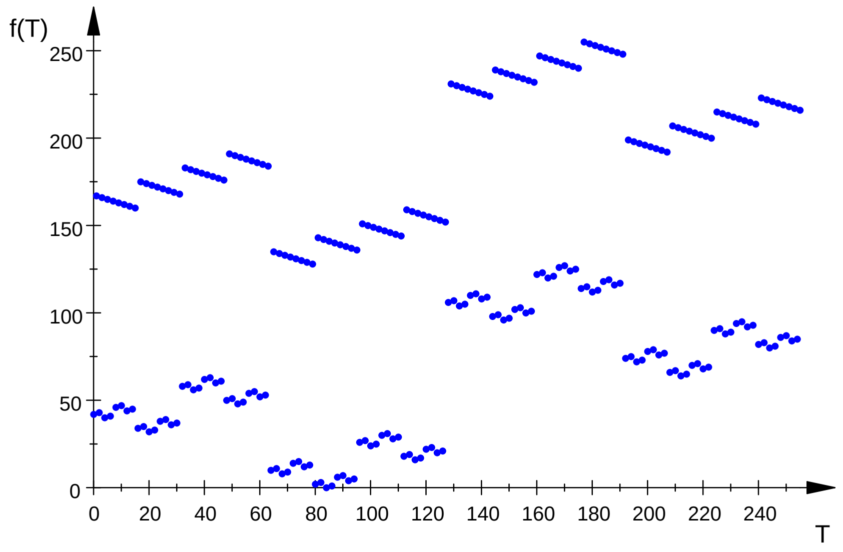

Figure 5.2: The line $f\colon \mathbb{F}_{256} \longrightarrow \mathbb{F}_{256},\ T \mapsto \left(x^7 + x^3 + x^2 + 1\right) T + \left(x^5 + x^3 + x\right)$ over the field $\mathbb{F}_{256}$ carries the secret $f(0) = x^5 + x^3 + x \mathrel{\widehat{=}} 2^5 + 2^3 + 2 = 42$ and passes through zero at $T \mathrel{\widehat{=}} 84$. The elements of $\mathbb{F}_{256}$ are represented as polynomials in x of degree less than 8 over $\mathbb{F}_2 = \mathbb{Z}_2$ modulo $x^8 + x^4 + x^3 + x + 1$ and identified with integers by 'evaluating' such a polynomial over the integers at $x = 2$.

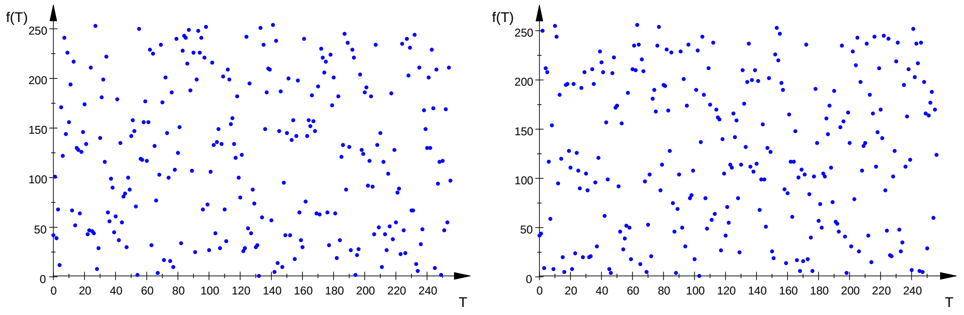

Figure 5.3: The cubic curve $f\colon \mathbb{F}_{256} \longrightarrow \mathbb{F}_{256},\ T \mapsto (x^7+x^6+x^5+x^4+x^3+x+1)\,T^3 + (x^7+x^5+x+1)\,T^2 + (x^4+x^2+x+1)\,T + (x^{\,5}+x^3+x)$ over $\mathbb{F}_{256}$ on the left hand side and $f\colon \mathbb{F}_{257} \longrightarrow \mathbb{F}_{257},\ T \mapsto 20\,T^3 + 42\,T^2 + (-60)\,T + 42$ over $\mathbb{F}_{257}$ on the right hand side each carry the secret $f(0) \mathrel{\widehat{=}} 42$. For our untrained eyes the nice structure of this curve is not visible but still: any four points determine the entire polynomial and thus the secret.

Depositing the coin is easy, too:

PROTOCOL 5.9. Deposit.

1. Martin sends the entire transcript of the payment Protocol 5.8 to the bank.
2. She then looks up the signature in her database.
   - ◦ If she does not find it, Martin gets his money put on his account and a receipt.
   - ◦ Otherwise, the bank detects a double spending just as in the other systems:
     - – If the challenges x and x' are also equal then Martin has tried to redeposit a coin.
     - – Otherwise the bank tries to reveal Alice' identity. For now the bank knows $r \equiv_v kx+U$ and $r' \equiv_v kx'+U$ modulo v which is just a linear system of equations for k and U. Now she can take Alice to court for double spending.

There are several points to be taken into account for the withdrawal process. Of course the first requirement is that the bank cannot link the withdrawal and the deposit of a coin (unless a double spending occurs). Further, it shall be guaranteed that the parameters a, b, c and k are chosen randomly. Both parties, in particular the bank in our case, have to be sure that these parameters are not 'made up'. To do so Alice and the bank each choose a part, say a' and a'' of these parameters and at the end they take the product $a = a'a''$. Only both must make their choice independently whereas we have no way of guaranteeing a parallel transmission of the respective shares. (Actually, this seems very similar to 'Coin flipping by phone', Blum 1982.) To achieve this, Alice first chooses a' and then transmits some information $\tilde{A}$ which binds her to this value of a'. Then the bank chooses a'' and sends it to Alice. Actually, in our case the product must only be known to Alice. To make sure that Alice continues as desired, Alice sends something which requires that she uses the bank's a'' in order to give her the desired meaningful signature. Or the bank's answer depends on the information $\tilde{A}$ that binds Alice. Then the answer is only useful to Alice if she sticks to her previously chosen value a'.

5.4.3. Randomized blind signatures. First we consider how to get a *randomized blind signature*. Randomized means that the bank will be sure that the used parameter was indeed chosen at random. Blind means, as usual, that the bank cannot link the final signature to the transcript of the signature protocol. And of course Alice should not be able to generate such a signature on her own (this makes it a signature). Thus this scheme will be well suited for our needs. Ferguson attributes it to Chaum (1992). Additionally we use a one-way hash function $f\colon \mathbb{Z}_N^{\times} \longrightarrow \mathbb{N}_{<v}$.

PROTOCOL 5.10. Randomized blind signature.

1. Alice randomly chooses $a', \alpha \in \mathbb{Z}_N^\times$ and $\sigma \in \mathbb{N}_{<v}$. She computes $\widetilde{A} \leftarrow \alpha^v a' g^\sigma$ and sends that to the bank. $\xrightarrow{\widetilde{A}}$
2. The bank randomly chooses $a'' \in \mathbb{Z}_N^\times$ and sends it to Alice. $\xleftarrow{a''}$
3. Alice computes $a \leftarrow a'a'' \in \mathbb{Z}_N^\times$ and an adjusting exponent $e + \widehat{e}v \leftarrow f(a) - \sigma$ with $e \in \mathbb{N}_{<v}$ and sends e to the bank. $\xrightarrow{e}$
4. The bank computes $\overline{A} \leftarrow \widetilde{A} \cdot a'' g^e$ and sends Alice a signature $\widetilde{S} \leftarrow \overline{A}^{(1/v)}$ of it. $\xleftarrow{\widetilde{S}}$
5. Alice unblinds the signature to obtain $S \leftarrow \widetilde{S}\alpha^{-1}g^{\widehat{e}}$. Now she has a signature pair (a, S) satisfying

$$S^v \overset{?}{=} a g^{f(a)}. \tag{5.11}$$

Before we discuss attacks let us have a short glance at the correctness. There is one complication that we did not mention in advance. Actually, Alice must hand over $e \in \mathbb{N}_{<v}$ instead of $e + \widehat{e}v$ in order to keep her secrets protected. Unfortunately, it is not allowed to calculate modulo v (or any other number Alice knows of) in the exponent of g. She only knows that g has large order but she has no idea which one. Thus she will obtain a v-th root of $ag^{f(a)-\widehat{e}v}$ instead of a v-th root of $ag^{f(a)}$. Luckily this is correctable since the deviation is a v-th power of a known value. Indeed, we have

$$\begin{aligned} S^v &= \widetilde{S}^v \alpha^{-v} g^{\widehat{e}v} \\ &= \overline{A}\alpha^{-v} g^{\widehat{e}v} \\ &= \widetilde{A} \cdot a'' g^e \alpha^{-v} g^{\widehat{e}v} \\ &= \alpha^v a' g^\sigma \cdot a'' \alpha^{-v} g^{f(a)-\sigma} \\ &= a g^{f(a)}. \end{aligned}$$

First, note the relations between the values in the transcript: Clearly, Step 4 in Protocol 5.10 implies

$$\widetilde{S}^v = \widetilde{A} \cdot a'' g^e. \tag{5.12}$$

Everything else in the transcript is independent, as we will see shortly. Indeed, even if Alice follows the protocol any combination of $\widetilde{A}$, a'' and e can occur: First choose any value for a, then solve $e + \widehat{e}v = f(a) - \sigma$ for $\sigma \in \mathbb{N}_{<v}$ and $\widehat{e}$, $a = a'a''$ for a'', and $\widetilde{A} = \alpha^v a' g^\sigma$ for α. (We do not care for efficiency here!) Thus (5.12) is the only relation. Each protocol transcript even occurs with the same probability. The only choice is the choice of a, all other solutions are unique. Thus in order to obtain a valid signature from the protocol Alice can choose $\widetilde{A}$ and e but must then go along with a'' and $\widetilde{S}$ as given by the bank. Though Alice can choose $\widetilde{A}$ as a v-th power of something she knows, her major problem is that she does not know the v-th root of a'' and thus cannot correct this factor to her needs without breaking RSA.

What if the bank tries to trace Alice? Can she get any information on the pair (a, S) that is Alice' signature at the end? No, she cannot. Indeed, each such pair occurs with the same probability from the view of the bank. The bank knows $\widetilde{A}$, a'', e and $\widetilde{S}$. Suppose Alice gets (a, S). Then there is exactly one choice for Alice that can have produced this outcome: $\sigma \in \mathbb{N}_{<v}$ and $\widehat{e}$ are uniquely determined by $e + \widehat{e}v = f(a) - \sigma$, α by $S = \widetilde{S}\alpha^{-1}g^{\widehat{e}}$, and a' by $a = a'a''$. The equation $\widetilde{A} = \alpha^v a' g^\sigma$ is implied by (5.12): $\widetilde{A} = \widetilde{S}^v \cdot (a'')^{-1} g^{-e} = \alpha^v S^v g^{\sigma - f(a)}/a'' = \alpha^v a g^{f(a)} g^{\sigma - f(a)}/a'' = \alpha^v a' g^\sigma$.

Let us see what happens if Alice tries to cheat. Clearly, she cannot solve (5.11) after fixing a unless she breaks the bank's signature which is assumed to be infeasible. But can she use the signature generation with a more or less prescribed a? As already stated only (5.12) binds the values of the transcript. Suppose she wants to get along with a prescribed a. What would she have to do in order to get a signature for it? To satisfy (5.12) she must solve $ag^{f(a)} = \widetilde{A} \cdot a'' g^e$ for e. She can choose $\widetilde{A}$ in a clever way, yet only before she knows a''. Writing $e + \widehat{e}v = f(a) - \sigma$ the equation $a = \widetilde{A} \cdot a'' g^{-\sigma - \widehat{e}v}$ must be solved for σ. Actually no matter how she has chosen $\widetilde{A}$ the task is to compute a discrete logarithm. But of course the parameters will be adjusted such that computing a discrete logarithm with base g is not feasible. By trying several σ at random she might get control of some bits of a but no more. Thus there seems at least to be no obvious way for Alice to cheat.

If Alice tries to use some more of the structure she might try to use some multiple of a power of (5.12) to obtain a valid signature on some expression $ag^{f(a)}$:

$$D\widetilde{S}^{Ev} = D\widetilde{A}^{E}(a'')^{E} \cdot g^{eE}$$

First note that D can only help if Alice knows a v-th root but that does not lead her far. To be helpful she might try to adjust this such that

$$\begin{aligned}(\widetilde{A}a'')^{E} &= ag^{t},\\ t + eE &= f(a)\end{aligned}$$

with some $t \in \mathbb{N}_{<v}$. Alice can use the first equation only after she knows a'', when $\widetilde{A}$ is already fixed. So the obvious way to solve these equations is to choose E and t and determine a by the first equation. The control over a she can obtain this way depends on her ability of computing discrete logarithms with respect to g or $\widetilde{A}a''$. Finally, the second equation determines e. However, that means that A, E and t must be chosen before e is transmitted.

Note that computing a discrete logarithm with base $\widetilde{A}a''$ might be feasible! If the order of $\widetilde{A}a''$ is smooth and can be determined efficiently then we can compute discrete logarithms efficiently and thus find a 'good' E. So we choose a, compute E and e. The order of the group $\mathbb{Z}_N^{\times}$ however is unknown to Alice and infeasible to find (unless she breaks RSA). The bank could adjust a'' a little to avoid very low order elements. Yet, this affects the distribution of a'' and might not be desirable. Probably, it is true anyway that most elements of $\mathbb{Z}_N^{\times}$ are difficult discrete logarithm bases provided $\varphi(N)$ contains large prime factors.

A way to stop Alice from even trying the just described manipulation is to change the scheme a little. In the previous 'attack', it was essential that Alice can compute $(a'')^{E}$. If we replace a'' by $h^{a''}$ then Alice cannot simply compute the corresponding $h^{((a'')^{E})}$ from $h^{a''}$. Since we compute $a = a'a''$ in $\mathbb{Z}_N$ the order of h must divide N. But we are not bound to the domains already in use and simply choose a prime t such that $h \in \mathbb{F}_t$ of order N exists, that is $t = \rho N + 1$ for some $\rho \in \mathbb{N}$. Once t is found any element $x \in \mathbb{F}_t^{\times}$ raised to the power $\frac{t-1}{N}$ gives an element $h = x^{\frac{t-1}{N}}$ of order 1, p, q, or N. The bank can easily exclude the first three cases by checking $h \neq 1$, $h^{p} \neq 1$ and $h^{q} \neq 1$. A drawback of this is that Alice cannot verify that. She is only able to check $h^{N} = 1$ and $h \neq 1$. But this is not really severe because it is in the bank's interest to have an element of highest possible order there. Of course we now have to modify the definition of the hash function, we need $f\colon \mathbb{F}_t^{\times} \to \mathbb{N}_{<v}$. In total we have the following

PROTOCOL 5.13. Randomized blind signature without exponential attack.

1. Alice randomly chooses $a', \alpha \in \mathbb{Z}_N^{*}$ and $\sigma \in \mathbb{N}_{<v}$. She computes $\widetilde{A} \leftarrow \alpha^{v}a'g^{\sigma}$ and sends that to the bank. $\xrightarrow{\widetilde{A}}$
2. The bank randomly chooses $a'' \in \mathbb{Z}_N^{\times}$, computes $\widetilde{h} \leftarrow h^{a''}$ and sends it to Alice. $\xleftarrow{\widetilde{h}}$
3. Alice computes an adjusting exponent $e + \widehat{e}v \leftarrow f(\widetilde{h}^{a'}) - \sigma$ with $e \in \mathbb{N}_{<v}$ and sends it to the bank. $\xrightarrow{e}$
4. The bank computes $\overline{A} \leftarrow \widetilde{A} \cdot a''g^{e}$ and sends Alice a signature $\widetilde{S} \leftarrow \overline{A}^{(1/v)}$ of it along with a''. $\xleftarrow{a'', \widetilde{S}}$
5. Alice calculates $a \leftarrow a'a''$ and unblinds the signature to obtain $S \leftarrow \widetilde{S}\alpha^{-1}g^{\widehat{e}}$. Now she has a signature pair (a, S) satisfying

(5.14) $$S^{v} \overset{?}{=} ag^{f(h^{a})}.$$

OPEN QUESTION 5.15. *Could Alice in either variant obtain more signatures than the number of times she executes the protocol?*

5.4.4. Withdrawal. For the withdrawal process we will use the previous signature scheme three times in parallel. Actually, for signing a we use the simple version and for signing b and c we use the one which is protected against the exponential attack. The first will be protected by an additional factor derived from the other two. As in the above protocols, Alice and the bank will each choose a share of the three values. Yet, a will be furthermore linked to the other two. This procedure would hand over three signatures to Alice. As we already saw in Protocol 5.8 which defined the payment from Alice to Martin, Alice needs signatures of AB^k and AC^U. Since we are using the RSA scheme to compute signatures these two are merely combinations of the three signatures to A, B and C.

The bank must be sure that U is used as specified since this is the identity coded into the coin. It will enable the bank to trace Alice in case of a double spending. This will be guaranteed since the bank puts together the second signature as one for AC^U.

It is in Alice' interest that k is randomly chosen and only known to herself since this parameter protects her identity! If it were known to anyone else then after only one payment U could be computed. But also the bank shall be sure that this parameter is chosen at random because otherwise Alice could try to fit this parameter according to her needs. Thus the bank will only hand over a signature to $A^{1/k'}C^{k''}$ without any knowledge of k' but with almighty power over k''. Using $A^{1/k'}$ (implicitly) is made possible by choosing a as a k'-th power. Alice can later raise the result to the k'-th power and thus giving her a signature of $AC^{k'k''}$ as desired.

One problem arises again several times: Alice has to correct the exponents that shall be dealt with only modulo v. For example, this happens to $k'k''$. The final exponent to be used must be $k = (k'k'') \operatorname{rem} v$. Since the difference is a multiple of v in some exponent Alice can correct that even in the v-th root. As can be verified in the protocol the corrections $\widehat{e_2}$ and $\widehat{e_3}$ are either 0 or -1. But the corrections $\widehat{1}$, $\widehat{e_1}$, and $\widehat{k}$ use the entire range $\mathbb{N}_{<v}$.

PROTOCOL 5.16. Withdrawal.

1. Alice chooses random shares $a', b', c' \in_R \mathbb{Z}_N^\times$, random blinding bases $\alpha, \beta, \gamma \in_R Z_N^\times$, and random blinding exponents $\sigma, \tau, \varphi \in_R \mathbb{N}_{<v}$. She computes the blinded candidates $\widetilde{A} \leftarrow \alpha^v a' \cdot g_1^\sigma$, $\widetilde{B} \leftarrow \beta^v b' \cdot g_2^\tau$, $\widetilde{C} \leftarrow \gamma^v c' \cdot g_3^\varphi$ and sends them to the bank. $\xrightarrow{\widetilde{A}, \widetilde{B}, \widetilde{C}}$
2. The bank chooses her random shares $a'', b'', c'' \in_R \mathbb{Z}_N^\times$ and sends a'', $\widetilde{h_2} \leftarrow h_2^{b''}$, $\widetilde{h_3} \leftarrow h_3^{c''}$ to Alice. $\xleftarrow{a'', \widetilde{h_2}, \widetilde{h_3}}$
3. Alice computes

$$e_2 + \widehat{e_2} v \leftarrow f_2(\widetilde{h_2}^{b'}) - \tau \qquad \text{with } e_2 \in \mathbb{N}_{<v},$$
$$e_3 + \widehat{e_3} v \leftarrow f_3(\widetilde{h_3}^{c'}) - \varphi \qquad \text{with } e_3 \in \mathbb{N}_{<v}.$$

   and chooses $k' \in_R \mathbb{N}_{<v}^\times$. After computing $a \leftarrow (a'a'' \cdot f_4(e_2, e_3))^{k'}$ and $k^- \in \mathbb{N}_{<v}$, $\widehat{1} \in \mathbb{N}$ such that $k'k^- = 1 + \widehat{1}v$, she computes

$$e_1 + \widehat{e_1} v \leftarrow k^- f_1(a) - \sigma \qquad \text{with } e_1 \in \mathbb{N}_{<v}.$$

   Then she sends the exponents (e_1, e_2, e_3) to the bank. $\xrightarrow{e_1, e_2, e_3}$
4. The bank computes $\overline{A} \leftarrow \widetilde{A} a'' f_4(e_2, e_3) g_1^{e_1}$, $\overline{B} \leftarrow \widetilde{B} b'' g_2^{e_2}$, $\overline{C} \leftarrow \widetilde{C} c'' g_3^{e_3}$. Then the bank chooses her share $k'' \in_R \mathbb{N}_{<v}^\times$ of k. She then computes the signatures $\widetilde{S}_1 \leftarrow (\overline{A}\,\overline{C}^{k''})^{(1/v)}$ and $\widetilde{S}_2 \leftarrow (\overline{B}\,\overline{C}^U)^{(1/v)}$ and sends them to Alice. $\xleftarrow{b'', c'', k'', \widetilde{S}_1, \widetilde{S}_2}$
5. Alice puts everything together: she computes $b \leftarrow b'b''$, $c \leftarrow c'c''$ in $\mathbb{Z}_N^\times$, and $k + \widehat{k} v \leftarrow k'k''$ with $k \in \mathbb{N}_{<v}$. Now she can compute

$$A \leftarrow a g_1^{f_1(a)}, \quad B \leftarrow b g_2^{f_2(h_2^b)}, \quad C \leftarrow c g_3^{f_3(h_3^c)}$$

   and unblind the signatures $S_1 \leftarrow \left(\widetilde{S}_1 \left(\alpha^{-1} g_1^{\widehat{e_1}}\right) \left(\gamma^{-1} g_3^{\widehat{e_3}}\right)^{k''}\right)^{k'} g_1^{-f_1(a)\widehat{1}} C^{-\widehat{k}}$ and $S_2 \leftarrow \widetilde{S}_2 \left(\beta^{-1} g_2^{\widehat{e_2}}\right) \left(\gamma^{-1} g_3^{\widehat{e_3}}\right)^U$. She now has a coin

(a, b, c, k, S_1, S_2) with the property

$$S_1^v = AC^k, \quad S_2^v = BC^U. \tag{5.17}$$

First we verify that this indeed fulfills the claimed equations (5.17). What the bank obtains actually is

$$\begin{aligned}
\overline{A}^{k'} &= A \cdot \left(\left(\alpha g_1^{-\widehat{e_1}}\right)^{k'} g_1^{f_1(a)\widehat{1}}\right)^v,\\
\overline{B} &= B \cdot (\beta g_2^{-\widehat{e_2}})^v,\\
\overline{C} &= C \cdot (\gamma g_3^{-\widehat{e_3}})^v.
\end{aligned}$$

With this information we can exploit the definitions:

$$\begin{aligned}
S_1^v &= \left(\left(\widetilde{S}_1 \left(\alpha^{-1} g_1^{\widehat{e_1}}\right)\left(\gamma^{-1} g_3^{\widehat{e_3}}\right)^{k''}\right)^{k'} g_1^{-f_1(a)\widehat{1}} C^{-\widehat{k}}\right)^v\\
&= \overline{A}^{k'} \left(\left(\alpha^{-1} g_1^{\widehat{e_1}}\right)^{k'} g_1^{-f_1(a)\widehat{1}}\right)^v \left(\overline{C}\left(\gamma^{-1} g_3^{\widehat{e_3}}\right)^v\right)^{k'k''} C^{-\widehat{k}v}\\
&= AC^{k'k''-\widehat{k}v} = AC^k
\end{aligned}$$

and similarly

$$\begin{aligned}
S_2^v &= \left(\widetilde{S}_2 \left(\beta^{-1} g_2^{\widehat{e_2}}\right)\left(\gamma^{-1} g_3^{\widehat{e_3}}\right)^U\right)^v\\
&= \overline{B}\left(\beta^{-1} g_2^{\widehat{e_2}}\right)^v \left(\overline{C}\left(\gamma^{-1} g_3^{\widehat{e_3}}\right)^v\right)^U\\
&= BC^U.
\end{aligned}$$

Thus the key equations (5.17) hold.

In order to prevent the bank from framing an innocent Alice for double spending at some time Alice must provide a signature for this identity U. If this is not the case not only Alice will not trust the system but also the bank will not be able to prosecute Alice for a potential double spending. No court would blame Alice if she *can* be framed by the bank. Yet, we must somehow guarantee this in the withdrawal process, see below. The first thought how to implement this is to make U a signed version of Alice' identity. But then the bank cannot directly control that U has the correct form and thus Ferguson suggests a different approach.

5.4.5. Summary. Ferguson (1994b) uses polynomial secret sharing to allow many possible queries. To embed a polynomial $kx + U$ into the system we proceed like this: Three numbers a, b, c are chosen at random by the bank and Alice. For the mutual security it is important that each partner is sure that these figures are indeed random. This is done by something similar to 'coin flipping by phone'. Alice and the bank each choose a part of each number and the actual number then is composed of these two parts. Yet only Alice will know the outcome of the random number. This makes the system anonymous. From these numbers are derived three numbers A, B, C with the help of some one-way functions. This ensures that Alice has almost no influence on the specific values of these three numbers. In the withdrawal process the bank sends Alice RSA signatures for AC^k and BC^U. Here also k must be a random quantity and again both must be sure of it.

To answer a query x by Martin Alice shows a signature R to $(AC^k)^x(BC^U) = A^x BC^{kx+U}$. She can produce this new signature from the two she knows. Clearly, she must also hand over $r = kx+U$ since Martin cannot compute this quantity. If the bank gets two such answers the bank can solve for k and U and thus reveal Alice' identity coded in U.

That is a very brief sketch of the system. There are some complications in the way the numbers a, b, c and k are chosen and some technical details that are used to prevent certain kinds of attacks.

6. Further topics

Still there are completely different threats.

6.1. The perfect crime. Complete anonymity also has its drawbacks:

- ○ Ed kidnaps a baby.
- ○ He prepares 10 000 money orders for 1 000 € each and blinds them.
- ○ He sends them to the authorities with the threat to kill the baby unless the following instructions are met: A bank signs all orders and the results are published in a newspaper (or by any other kind of broadcast).
- ○ The authorities comply.
- ○ Ed buys a newspaper, unblinds the orders and spends the coins. There is no way for the authorities to trace the money to him.
- ○ Ed frees the baby.

6.2. Further properties. Fairness, re-usability, . . .

6.3. 'Historical' remarks. DigiCash(David Chaum), eCash/Cybercash(Hettinga), flooz, . . . To my knowledge no electronic cash system is used in practice.

6.4. Social aspects. Acceptance, necessity, environment.

6.5. Economical aspects. Transaction costs, existing concurrent paying systems.

7. Texts on electronic money

For the great congress several texts describing typically one system for electronic money have been considered. Finally, those marked with a circle '○' have been used.

- – Chaum (1985) (surface description using lots of pictograms),
- ○ Chaum, Fiat & Naor (1989) (unconditionally untraceable/anonymous electronic cash system [DigiCash]; no proofs),
- ○ Ferguson (1993, 1994b) (nicely described protocols including some reasoning),
- – Ferguson (1994a) (extensions to Ferguson (1993, 1994b): multi-spendable coins, observers),
- – Brands (1993) (extensive paper containing Brands (1994a,b, 1995), section 11, 12 describe the basic system, sections 9, 10 are needed for details, sections 5, 6, 8 concern the underlying representation problem),
- – Brands (1994b) (with observers, complete protocols),
- – Brands (1994a) (with smart cards, coins and signatures are separated, uses Schnorr signatures, very small memory requirements),
- – Brands (1995) (off-line, no observer or smart card needed, uses Schnorr like signatures),
- ○ Brands (1999), first four pages and section 4 (overview article, section 4 contains an example system similar (or equal?) to Brands (1995), section 2 describes preliminaries: modeling electronic cash, authentication techniques, [conventional dynamic authentication; dynamic authentication based on public key cryptography], section 3 dwells on electronic cash techniques: representing electronic cash, transferring electronic cash [transferring register based electronic cash; transferring electronic coins], when tamper-resistance is compromised [fraud detection; fraud tracing; fraud liability; fraud containment], security for account holders [preventing loss, preventing payment redirection, non-repudiation], privacy of payments [relaxed monitoring, anonymous accounts and anonymous devices; blinding; one-show blinding; guaranteeing your own privacy; one-sided versus two-sided untraceability]),

– Medvinsky & Neuman (1993) (NetCash is a system to make various forms of electronic money be exchangeable and acceptable via various channels),

– Frankel, Tsiounis & Yung (1996, 1998) (fair electronic cash),

– Bellare, Garay, Jutla & Yung (1998a,b) (),

– Maitland & Boyd (2001) (use group signatures),

○ Schneier (1996), §6.4 (describes Chaum's system, advancing step by step)

– Kou (2003), §8.3 (describes Brands' system, probably Brands (1995)),

– Kou (2003), §8.4 (one-response digital cash),

○ Kou (2003), §8.5 (fair digital cash).

References

MANINDRA AGRAWAL, NEERAJ KAYAL & NITIN SAXENA (2003). PRIMES is in P. URL `http://www.cse.iitk.ac.in/news/primality_v3.pdf`. Preprint, v3.

M. BELLARE, J. GARAY, C. JUTLA & M. YUNG (1998a). VarietyCash: a Multi-purpose Electronic Payment System. In *Proceedings of the 3rd Usenix Workshop on Electronic Commerce*, BENNET S. YEE, editor, 17 pages. URL `http://www.usenix.org/events/ec98/bellare.html`. Full version is Bellare *et al.* (1998b).

M. BELLARE, J. GARAY, C. JUTLA & M. YUNG (1998b). VarietyCash: a Multi-purpose Electronic Payment System. URL `http://www-cse.ucsd.edu/users/mihir/papers/vc.html`. Proceedings version see Bellare *et al.* (1998a).

MANUEL BLUM (1982). Coin flipping by telephone. In *CRYPTO 1981*, 133–137. IEEE. ISSN 0000-0133. URL `http://www-2.cs.cmu.edu/~mblum/research/pdf/coin/`.

STEFAN A. BRANDS (1993). An Efficient Off-line Electronic Cash System Based On The Representation Problem. Technical report, Centrum voor Wiskunde en Informatica (CWI). URL `http://citeseer.nj.nec.com/brands93efficient.html`.

STEFAN BRANDS (1994a). Off-Line Cash Transfer by Smart Cards. Technical Report CS-R9455 1994, Centrum voor Wiskunde en Informatica. URL `http://citeseer.nj.nec.com/brands94offline.html`.

STEFAN BRANDS (1994b). Untraceable Off-Line Cash in Wallets with Observers. In *Advances in Cryptology: Proceedings of CRYPTO '93*, Santa Barbara CA, DOUGLAS R. STINSON, editor, number 773 in Lecture Notes in Computer Science. Springer-Verlag, New York. ISBN 0-387-57766-1. ISSN 0302-9743. URL `http://link.springer.de/link/service/series/0558/bibs/0773/07730302.htm`.

STEFAN BRANDS (1995). Off-Line Electronic Cash Based on Secret-Key Certificates. In *Proceedings of LATIN '95*, Valparaíso, Chile. Valparasio, Chili. URL `http://citeseer.nj.nec.com/brands95offline.html`.

STEFAN BRANDS (1999). Electronic Cash. In *Handbook on Algorithms and Theory of Computation*, MIKHAIL J. ATALLAH, editor, chapter 44. CRC Press, Boca Raton. ISBN 0-8493-2649-4. URL `http://citeseer.ist.psu.edu/brands98electronic.html`.

DAVID CHAUM (1985). Security without identification: transaction systems to make big brother obsolete. *Communications of the ACM* **28**(10), 1030–1044. ISSN 0001-0782. URL `http://portal.acm.org/citation.cfm?id=4373&coll=portal&dl=ACM&CFID=41194678&CFTOKEN=68326228`.

DAVID CHAUM (1992). Randomized blind signature. Personal communication with Niels Ferguson.

DAVID CHAUM, AMOS FIAT & MONI NAOR (1989). Untraceable Electronic Cash (Extended Abstract). In *Advances in Cryptology: Proceedings of CRYPTO '88*, Santa Barbara CA. URL `http://citeseer.nj.nec.com/chaum89untraceable.html`.

C. C. COCKS (1973). A note on 'non-secret encryption'. CESG Memo. URL `http://www.cesg.gov.uk/site/publications/media/notense.pdf`.

WHITFIELD DIFFIE & MARTIN E. HELLMAN (1976). New directions in cryptography. *IEEE Transactions on Information Theory* **IT-22**(6), 644–654.

J. H. ELLIS (1970). The possibility of secure non-secret digital encryption.

J. H. ELLIS (1987). The history of Non-Secret Encryption. Communications-Electronics Security Group, part of the United Kingdom Civil Service (CESG). URL `http://www.cesg.gov.uk/site/publications/media/ellis.pdf`.

NIELS FERGUSON (1993). Single Term Off-Line Coins. Technical Report CS-R9318, Centrum voor Wiskunde en Informatica, Amsterdam. URL `http://www.macfergus.com/pub/CS-R9318.html`.

NIELS FERGUSON (1994a). Extensions of Single Term Coins. In *Advances in Cryptology: Proceedings of CRYPTO '93*, Santa Barbara CA, DOUGLAS R. STINSON, editor, number 773 in Lecture Notes in Computer Science, 292–301. Springer-Verlag, New York. ISBN 0-387-57766-1. ISSN 0302-9743. URL `http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=773&spage=292`.

NIELS FERGUSON (1994b). Single Term Off-Line Coins. In *Advances in Cryptology: Proceedings of EUROCRYPT 1993*, Lofthus, Norway, TOR HELLESETH, editor, number 765 in Lecture Notes in Computer Science, 318–328. Springer-Verlag, Heidelberg. ISBN 3-540-57600-2. ISSN 0302-9743. URL `http://www.springerlink.com/openurl.asp?genre=article&issn=0302-9743&volume=765&spage=318`.

YAIR FRANKEL, YIANNIS TSIOUNIS & MOTI YUNG (1996). "Indirect Discourse Proofs": Achieving Efficient Fair Off-Line E-Cash. In *Advances in Cryptology - ASIACRYPT '96*, K. KIM & T. MATSUMOTO, editors, number 1163 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg. ISBN 3-540-61872-4. ISSN 0302-9743. Also at `http://citeseer.nj.nec.com/frankel96indirect.html` and `http://www.ccs.neu.edu/home/yiannis/papers/folc.ps.Z`.

YAIR FRANKEL, YIANNIS TSIOUNIS & MOTI YUNG (1998). Fair Off-Line e-Cash made easy. In *Advances in Cryptology - ASIACRYPT'98*, K. OHTA & D. PEI, editors, number 1514 in Lecture Notes in Computer Science. Springer-Verlag, Berlin, Heidelberg. ISBN 3-540-65109-8. ISSN 0302-9743. Also available at `http://citeseer.nj.nec.com/frankel98fair.html` and `http://www.ccs.neu.edu/home/yiannis/papers/folc-es.ps`.

JOACHIM VON ZUR GATHEN & JÜRGEN GERHARD (2003). *Modern Computer Algebra*. Cambridge University Press, Cambridge, UK, 2nd edition. ISBN 0-521-82646-2, 800. URL `http://cosec.bit.uni-bonn.de/science/mca.html`. First edition 1999.

WEIDONG KOU (editor) (2003). *Payment Technologies for E-Commerce*. Springer-Verlag, Berlin, Heidelberg, New York. ISBN 3-540-44007-0, X, 334 pp. URL `http://www.springeronline.com/3-540-44007-0`.

GREG MAITLAND & COLIN BOYD (2001). Fair Electronic Cash Based on a Group Signature Scheme. In *ICICS 2001*. URL `http://sky.fit.qut.edu.au/~boydc/papers/GOC-ECash.pdf`.

GENNADY MEDVINSKY & B. CLIFFORD NEUMAN (1993). NetCash: A design for practical electronic currency on the Internet. In *CCS'93: Proceedings of the First ACM Conference on Computer and Communications Security, November 1993, Fairfax, Virginia, United States, November 03-05, 1993.*, DOROTHY DENNING, RAY PYLE, RAVI GANESAN, RAVI SANDHU & VICTORIA ASHBY, editors, 102–106. ACM Press, New York. ISBN 0-89791-629-8. URL `http://portal.acm.org/citation.cfm?id=168588.168601`. See also `http://citeseer.ist.psu.edu/medvinsky93netcash.html`.

GENNADY MEDVINSKY & B. CLIFFORD NEUMAN (1994). Electronic Currency for the Internet. *ConneXions: the interoperability report* **8**(6), 19–23. ISSN 0894-5926. Also appeared as **?**.

R. L. RIVEST, A. SHAMIR & L. M. ADLEMAN (1978). A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Communications of the ACM* **21**(2), 120–126.

BRUCE SCHNEIER (1996). *Applied cryptography : protocols, algorithms, and source code in C*. John Wiley & Sons, New York, 2nd edition. ISBN 0-471-12845-7, 0-471-11709-9, XXIII, 758.

A. SCHÖNHAGE & V. STRASSEN (1971). Schnelle Multiplikation großer Zahlen. *Computing* **7**, 281–292.

M. J. WILLIAMSON (1974). Non-secret encryption using a finite field. CESG Memo. URL `http://www.cesg.gov.uk/site/publications/media/secenc.pdf`.

M. J. WILLIAMSON (1976). Thoughts on cheaper non-secret encryption. CESG Memo. URL `http://www.cesg.gov.uk/site/publications/media/cheapnse.pdf`.

MICHAEL NÜSKEN
b-it (Bonn-Aachen International Center for Information Technology)
Dahlmannstr. 2
D53113 Bonn
Germany
`nuesken@bit.uni-bonn.de`
`http://cosec.bit.uni-bonn.de/cosec/members/nuesken/`