The following are important requirements for digital signatures.

- The signature must be tightly attached to the signed document.

- It should be easy to *sign* for the legitimate signer, easy to *verify* the signature for the recipient, and hard to *forge* a signature.

- The signer should not be able to deny that he signed the document.

- Sometimes it is important that a signed document can only be used once for its legitimate purpose, not several times (say, a cheque).
We can use any public key encryption scheme (set-up, keygen, enc, dec) to obtain a signature scheme, by simply reversing encryption and decryption.

\[
x = \text{dec}_{sk}(m)
\]

\[
y = \text{enc}_{pk}(x); \text{ verify: } m \overset{?}{=} y
\]

\[
\text{ver}_{pk}(m, z) = \text{“true”} \iff m = \text{enc}_{pk}(z).
\]
**Protocol.** ElGamal signature scheme.

Set-up.

Input: a security parameter $n$ given in unary.
Output: as below.

1. A cyclic group $G = \langle g \rangle$ with $d = \#G$ elements, where $d$ is an $n$-bit number. We also have an injective encoding function $G \rightarrow \mathbb{Z}_d$, denoted as $x \mapsto x^*$, which is easy to compute but otherwise has no particular properties. All these data are published.
**Protocol.** ElGamal signature scheme.

**Key generation.**

Output: secret and public keys.

1. Secret key \( sk = a \leftarrow \mathbb{Z}_d \) and public key \( pk = A = g^a \in G \).
**Protocol.** ElGamal signature scheme.

**Signing.**

Input: a message \( m \in \mathbb{Z}_d \).
Output: a signature \( \text{sig}_{sk}(m) \in G \times \mathbb{Z}_d \) of \( m \).

1. Choose a secret session key \( k \leftarrow \mathbb{Z}_d^\times \).
2. \( K \leftarrow g^k \in G \).
3. Calculate \( b \leftarrow k^{-1} \cdot (m - aK^*) \) in \( \mathbb{Z}_d \), where \( k^{-1} \) is the inverse of \( k \) in \( \mathbb{Z}_d \).
4. Transmit \( m \) and its signature \( \text{sig}_{sk}(m) = (K, b) \).

Verifying.

Input: message $m$ and a pair $(z, c) \in G \times \mathbb{Z}_d$.
Output: $\text{ver}_{pk}(m, z, c)$, which is either “true” or “false”.

1. Compute $u \leftarrow g^m$ and $v \leftarrow A^{z^*} z^c$ in $G$.
2. If $z \neq 1$ and $u = v$ then return “true” else return “false”. 
\[ A = g^a \]

\[ K = g^k \]

\[ b = k^{-1}(m - aK^*) \]

Bob: \[ m, (K, b) \] Alice: \[ g^m \overset{?}{=} A^{K^*}K^b \]
Example

Bob has set up the publicly known group $G = \mathbb{Z}_{17}^\times$, with order $d = 16$, $g = 3$, and the “identity” mapping $\ast$ as above. Now he chooses his secret key, say $sk = a = 9$, and publishes $pk = A = 3^9 = 14$ in $\mathbb{Z}_{17}^\times$. Suppose that he wants to sign $m = 11$ and chooses $k = 5$ as secret session key. Then indeed $\text{gcd}(5, 17 - 1) = 1$, and $k^{-1} = -3 = 13$ in $\mathbb{Z}_{16}$. He calculates

$$K = 3^5 = 5 \text{ in } \mathbb{Z}_{17},$$

$$b = 13 \cdot (11 - 9 \cdot 5) = 6 \text{ in } \mathbb{Z}_{16},$$

since $K^* = K = 5$. Bob sends the message 11 together with its signature $\text{sig}_9(11) = (5, 6)$. Alice checks that $5 \neq 1$ and computes

$$u = g^m = 3^{11} = 7 \text{ in } \mathbb{Z}_{17},$$

$$v = A^{K^*} K^b = 14^5 \cdot 5^6 = 7 \text{ in } \mathbb{Z}_{17},$$

and accepts the message as properly signed.
Lemma

Let $d$ be a prime number. Then the verification procedure works correctly as specified, and the signature scheme can be implemented efficiently.
Schnorr takes for his signature scheme a large prime $p$ and the large group $\mathbb{Z}_p^\times$; this is what ElGamal used. But now Schnorr chooses a fairly small prime divisor $d$ of $p - 1 = \#\mathbb{Z}_p^\times$ with $\ell$ bits, and a subgroup $G \subseteq \mathbb{Z}_p^\times$ of order $d$. Then we have index calculus attacks on $\mathbb{Z}_p^\times$ or generic methods for $G$; nothing better is known. The latter take time $\Omega(\sqrt{d})$. 

Set-up.

Input: two security parameters $\ell < n$ in unary.

1. Choose an $\ell$-bit prime $d$ and an $n$-bit prime $p$ with $d$ dividing $p - 1$, a generator $g$ of a group $G = \langle g \rangle \subseteq \mathbb{Z}_p^\times$ of order $d$, and a map $*$ from $\mathbb{Z}_p^\times$ to $\mathbb{Z}_d$. 
**Protocol.** Schnorr signature scheme (DSA).

**Key Generation.**

**Output:** secret and public keys.

1. Secret key $sk = a \leftarrow \mathbb{Z}_d$ and public key $pk = A \leftarrow g^a \in G$. 
**Protocol.** Schnorr signature scheme (DSA).

**Signing.**

Input: a message $m \in \mathbb{Z}_d$.
Output: a signature $\text{sig}_{sk}(m) \in G \times \mathbb{Z}_d$ of $m$.

1. Choose a secret session key $k \leftarrow \mathbb{Z}_d^\times$.
2. $K \leftarrow g^k \in G$.
3. Calculate $b \leftarrow k^{-1} \cdot (m - aK^*)$ in $\mathbb{Z}_d$, where $k^{-1}$ is the inverse of $k$ in $\mathbb{Z}_d$.
4. Transmit $m$ and its signature $\text{sig}_{sk}(m) = (K, b)$.

Verifying.

Input: message $m$ and a pair $(z, c) \in G \times \mathbb{Z}_d$.
Output: $\text{ver}_{pk}(m, z, c)$, which is either “true” or “false”.

1. Compute $u \leftarrow g^m$ and $v \leftarrow A^{z^*} z^c$ in $G$.
2. If $z \neq 1$, $z^d = 1$, and $u = v$ then return “true” else return “false”.
Figure: Tradeoff between $\ell$-bit generic discrete logarithms and $n$-bit index calculus.
<table>
<thead>
<tr>
<th>cost $A$ of arithmetic</th>
<th>ElGamal</th>
<th>DSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>cost of $g^k$</td>
<td>$(2048)^2$</td>
<td>$(2048)^2$</td>
</tr>
<tr>
<td>message length</td>
<td>$A \log_2 p \approx A \cdot 2048$</td>
<td>$A \log_2 d \approx A \cdot 200$</td>
</tr>
<tr>
<td>signature length</td>
<td>2048 bits</td>
<td>200 bits</td>
</tr>
<tr>
<td>attack</td>
<td>4096 bits</td>
<td>400 bits (!)</td>
</tr>
<tr>
<td></td>
<td>$DL$ in $\mathbb{Z}_p^\times$</td>
<td>$DL$ in $\mathbb{Z}_p^\times$ or in $G$</td>
</tr>
</tbody>
</table>