Diffie-Hellman key-exchange is an anonymous key-exchange protocol: neither side knows who they are talking to. To be vulnerable to a "man-in-the-middle" attack, Alice sends $g^x$ to Bob, who responds with $g^y$, where $g$ is a generator of a large cyclic group. Alice does not know that Eve is intercepting the messages. Eve can now compute $g^{xy}$ and present as the key to both Alice and Bob. Alice and Bob now have no idea!

What we require: authenticated key-exchange (not anonymous) and relies on a root of trust (e.g., a certificate authority).

On the web, one of the parties will authenticate themself by presenting a certificate.

To build authenticated key-exchange, we require more ingredients — namely, an integrity mechanism (e.g., a way to bind a message to a sender — a "public-key MAC" or digital signature)

We will revisit when discussing the TLS protocol.

Digital signature scheme: Consists of three algorithms:

- Setup $(1^n) \rightarrow (vk, sk)$: Output a verification key $vk$ and a signing key $sk$.
- Sign $(sk, m) \rightarrow \sigma$: Takes the signing key $sk$ and a message $m$ and outputs a signature $\sigma$.
- Verify $(vk, m, \sigma) \rightarrow 0/1$: Takes the verification key $vk$, a message $m$, and a signature $\sigma$, and outputs a bit $0/1$.

Two requirements:

- Correctness: For all messages $m \in M$, $(vk, sk) \leftarrow \text{KeyGen}(1^n)$, then $\Pr[\text{Verify}(vk, m, \text{Sign}(sk, m)) = 1] = 1$. [Honesty-generated signatures always verify]
- Unforgeability: Very similar to MAC security. For all efficient adversaries $A$, $\text{Sign}_{sk} = \Pr[A(1^n) = m] = \text{negl}(n)$, where $W$ is the output of the following experiment:

```
\begin{array}{c|c}
| adversary | challenger |
\hline
| vk         | (vk, sk) \leftarrow \text{KeyGen}(1^n) |
| m \in M    | (vk, sk) \leftarrow \text{KeyGen}(1^n) |
| \sigma \leftarrow \text{Sign}(sk, m) | |
| (m', \sigma') | |
\end{array}
```

Let $m_1, ..., m_q$ be the signing queries the adversary submits to the challenger. Then, $W = 1$ if and only if:

$\Pr[\text{Verify}(vk, m, \sigma) = 1 \text{ and } m \not\in \{m_1, ..., m_q\}] = \text{negl}(n)$

An adversary cannot produce a valid signature on a new message.

Exact analog of a MAC (slightly weaker unforgeability: require adversary to not be able to forge signature on new message)

$\Rightarrow$ MAC security required that no forgery is possible on any message. [needed for authenticated encryption]

$\Rightarrow$ digital signature algorithms, elliptic-curve cryptography, RSA

It is possible to build digital signatures from discrete log based assumptions (DSA, ECDSA)

But construction not intuitive until we see zero knowledge proofs

We will first construct from RSA (triple permutation)
We will now introduce some facts on composite-order groups:

Let \( N = p q \) be a product of two primes \( p, q \). Then, \( \mathbb{Z}_N = \{0, 1, \ldots, N-1\} \) is the additive group of integers modulo \( N \). Let \( \mathbb{Z}_N^* \) be the set of integers that are invertible (under multiplication) modulo \( N \).

\[
\chi \in \mathbb{Z}_N^* \text{ if and only if } \gcd(N, \chi) = 1
\]

Since \( N = p q \) and \( p, q \) are prime, \( \gcd(N, \chi) = 1 \) unless \( \chi \) is a multiple of \( p \) or \( q \):

\[
\mathbb{Z}_N^* = \{ p - q, 1 = p q - q + 1 = (p-1)(q-1) = \phi(N)
\]

Recall Lagrange's Theorem:

for all \( \chi \in \mathbb{Z}_N^* : \chi^{\phi(N)} = 1 \pmod{N} \quad \text{called Euler's theorem, but special case of Lagrange's theorem}

\[
\text{Euler's phi function}
\]

Hard problems in composite-order groups:

- Factoring: given \( N = p q \) where \( p \) and \( q \) are sampled from a suitable distribution over primes, output \( p, q \)
- Computing cube roots: Sample random \( \chi \in \mathbb{Z}_N^* \). Given \( y = \chi^3 \pmod{N} \), compute \( \chi \pmod{N} \)

\[
\Rightarrow \text{This problem is easy in } \mathbb{Z}_p^* \text{ when } 3 \nmid p - 1. \text{ Namely, compute } 3^{\chi} \pmod{p}, \text{ say using Euclid's algorithm, and then compute } y^{3^{\chi}} \pmod{p} = (\chi^3)^{\chi} \pmod{p} = \chi \pmod{p}.
\]

\[
\Rightarrow \text{Why does this procedure not work in } \mathbb{Z}_N^* \text{? Above procedure relies on computing } 3^{\chi} \pmod{\mathbb{Z}_N^*} = 3^{\chi} \pmod{\phi(N)}
\]

But we do not know \( \phi(N) \) and computing \( \phi(N) \) is as hard as factoring \( N \). In particular, if we know \( N \) and \( \phi(N) \), then we can write

\[
\left\{
\begin{array}{l}
N = p q \\
\phi(N) = (p-1)(q-1)
\end{array}
\right.
\]

[both relations hold over the integers]

and solve this system of equations over the integers (and recover \( p, q \))

Hardness of computing cube roots is the basis of the RSA assumption:

\[
\text{distribution over prime numbers}
\]

RSA assumption:

Take \( p, q \in \text{Primes}(1^n) \), and set \( N = p q \). Then, for all efficient adversaries \( A \),

\[
\Pr \left[ \chi \in \mathbb{Z}_N^* ; y \leftarrow A(N, \chi) : y^3 = x \right] = \neg g(1^n)
\]

more generally, can replace \( 3 \) with any \( e \) where \( \gcd(e, \phi(N)) = 1 \)

common choices:

\[
e = 3
\]

\[
e = 65537
\]

Hardness of factoring / RSA assumption:

- Best attack based on general number field sieve (GNFS) — runs in time \( \tilde{O}(N^{1/3}) \)

(some algorithm used to break discrete log over \( \mathbb{Z}_p^* \))

- For 112 bits of security, use RSA-2048 (\( N \) is product of two 1024-bit primes)

- 128 bits of security, use RSA-3072

- Both prime factors should have similar bit length (ECM algorithm factors in time that scales with smaller factor)

\[\Rightarrow \text{large key-sizes and computational cost} \Rightarrow \text{ECC generally preferred over RSA}\]
RSA problem gives an instantiation of more general notion called a trapdoor permutation:
\[ \text{FRSA} : \mathbb{Z}_N^* \rightarrow \mathbb{Z}_N^* \]
\[ \text{FRSA} (x) := x^e \pmod{N} \quad \text{where} \quad \gcd(N, e) = 1 \]
Given \( \varphi(n) \), we can compute \( d = e^{-1} \pmod{\varphi(n)} \). Observe that given \( d \), we can invert \( \text{FRSA} \):
\[ \text{FRSA}^{-1} (x) := x^d \pmod{N} \]
Then, for all \( x \in \mathbb{Z}_N^* \):
\[ \text{FRSA} (\text{FRSA}^{-1} (x)) = (\text{FRSA}(x))^d = x^{ed} \pmod{\varphi(n)} = x^1 = x \pmod{N} \]