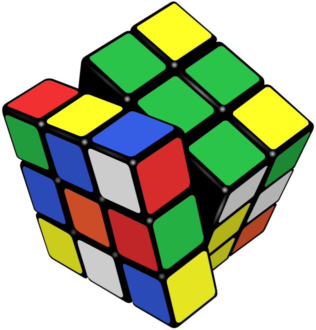
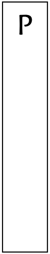
# Cryptography in Cyclic Groups (cont'd)



## Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.



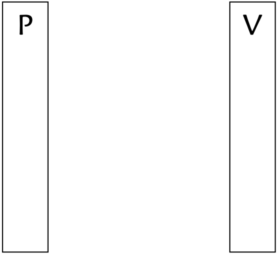

**Scenario**

$P$ sends $r = g^k$ where
$k \xleftarrow{\$} \mathbb{Z}_q$
$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$
$P$ sends $s = k + cx \bmod q$
$V$ checks whether
$g^s \cdot y^{-c} = r$

# Schnorr's ID Protocol (91)

## Let $\langle g \rangle$ be a group of prime order $q$

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.




### Scenario

$P$ sends $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

$P$ sends $s = k + cx \bmod q$

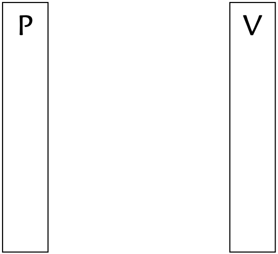
$V$ checks whether $g^s \cdot y^{-c} = r$

## Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.

$x \xleftarrow{\$} \mathbb{Z}_q$




**Scenario**

$P$ sends $r = g^k$ where

$k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

$P$ sends $s = k + cx \bmod q$

$V$ checks whether

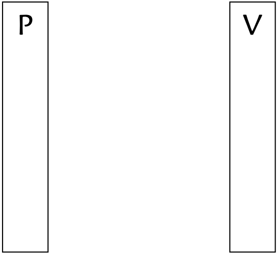
$g^s \cdot y^{-c} = r$

## Let $\langle g \rangle$ be a group of prime order $q$

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.

$x \xleftarrow{\$} \mathbb{Z}_q$

$y = g^x$




### Scenario

$P$ sends $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

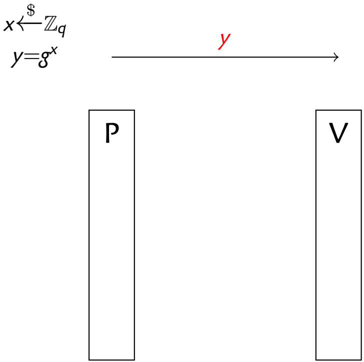
$P$ sends $s = k + cx \bmod q$

$V$ checks whether $g^s \cdot y^{-c} = r$

## Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.




**Scenario**

$P$ sends $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

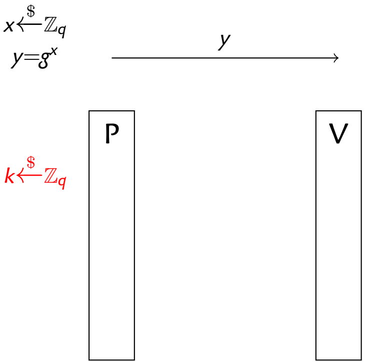
$P$ sends $s = k + cx \mod q$

$V$ checks whether $g^s \cdot y^{-c} = r$

## Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.




**Scenario**

$P$ sends $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

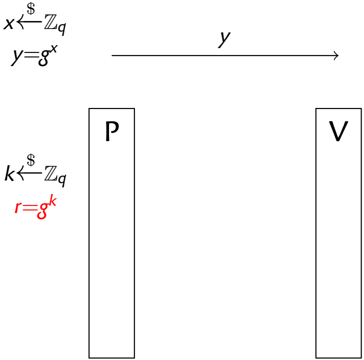
$P$ sends $s = k + cx \bmod q$

$V$ checks whether $g^s \cdot y^{-c} = r$

# Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.




**Scenario**

$P$ sends $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

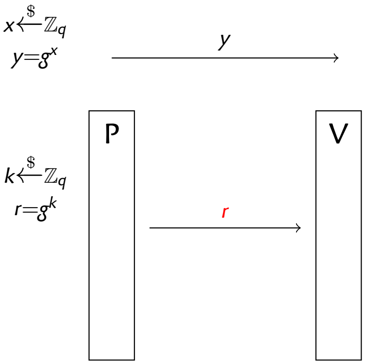
$P$ sends $s = k + cx \bmod q$

$V$ checks whether $g^s \cdot y^{-c} = r$

## Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a 3-move protocol.




**Scenario**

$P$ sends $r = g^k$ where

$k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

$P$ sends $s = k + cx \bmod q$

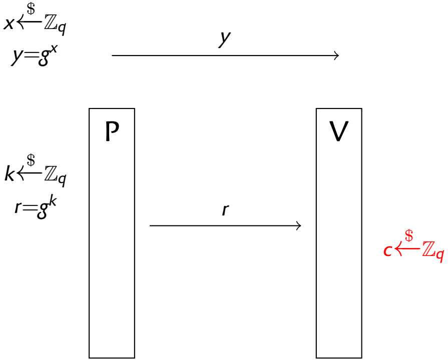
$V$ checks whether

$g^s \cdot y^{-c} = r$

## Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.




**Scenario**

$P$ sends $r = g^k$ where

$k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

$P$ sends $s = k + cx \bmod q$

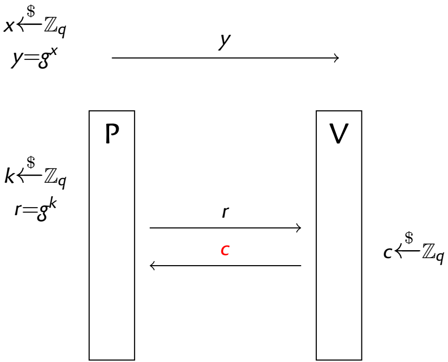
$V$ checks whether

$g^s \cdot y^{-c} = r$

# Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.




**Scenario**

$P$ sends $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

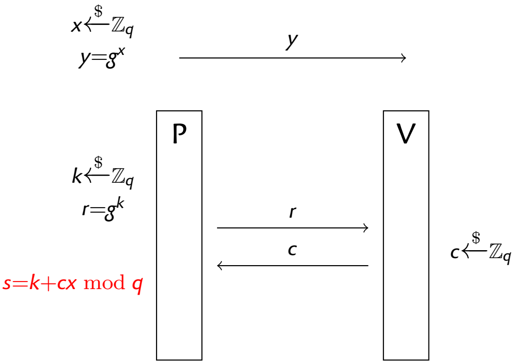
$P$ sends $s = k + cx \bmod q$

$V$ checks whether $g^s \cdot y^{-c} = r$

# Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.




**Scenario**

$P$ sends $r = g^k$ where

$k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

$P$ sends $s = k + cx \bmod q$

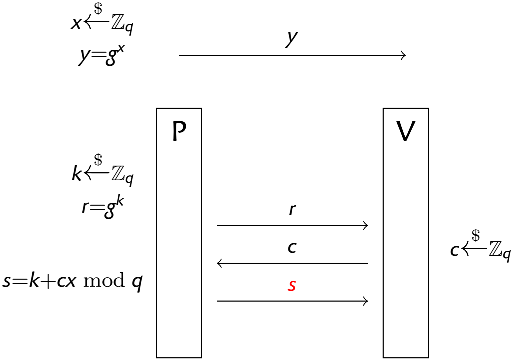
$V$ checks whether

$g^s \cdot y^{-c} = r$

## Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.




**Scenario**

$P$ sends $r = g^k$ where

$k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

$P$ sends $s = k + cx \bmod q$

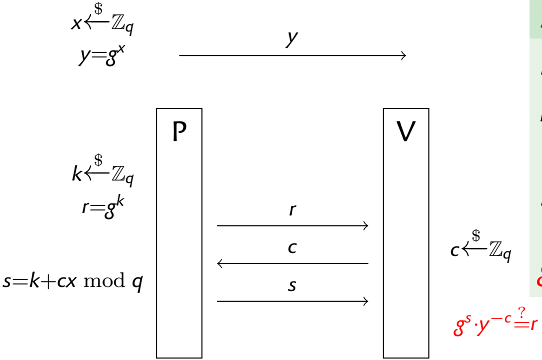
$V$ checks whether

$g^s \cdot y^{-c} = r$

## Schnorr's ID Protocol (91)

**Let $\langle g \rangle$ be a group of prime order $q$**

Prover $P$ proves to verifier $V$ that she knows the discrete log $x$ of a public group element $y = g^x$. It is a $3$-move protocol.




**Scenario**

$P$ sends $r = g^k$ where

$k \xleftarrow{\$} \mathbb{Z}_q$

$V$ sends $c \xleftarrow{\$} \mathbb{Z}_q$

$P$ sends $s = k + cx \mod q$

$V$ checks whether

$g^s \cdot y^{-c} = r$

## The Fiat-Shamir heuristic

**Fiat, Shamir (1986)**

How to Prove Yourself: Practical Solutions to Identification and Signature Problems.
Advances in Cryptology - Crypto'86, Lect. Notes Comput. Science 263, pp. 186-194.

- In such a $3$-pass identification scheme, the messages are called **commitment**, **challenge** and **response**. The challenge is randomly chosen by $V$.

**Fiat-Shamir Transform**

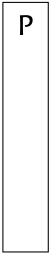
Replace the challenge by a hash value taken on scheme parameters and $t$, thereby removing $V$. This transforms the protocol by making it *non-interactive*.

The intuition is that any "sufficiently random" hash function should preserve the security of the protocol.

## Schnorr Signatures (via the Fiat-Shamir Transform)

**Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$**

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




**Signing and Verifying**

$\textsc{Sign}$

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \mod q$
$P$ sends $\sigma = (s, c)$

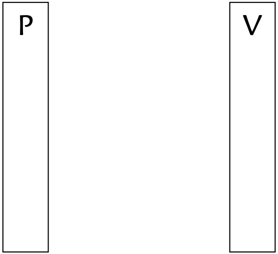
$\textsc{Ver}$

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

# Schnorr Signatures (via the Fiat-Shamir Transform)

## Introduce a hash function $H : \{0,1\}^{\star} \mapsto \mathbb{Z}_q$

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




## Signing and Verifying

$\textsc{Sign}$

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \mod q$
$P$ sends $\sigma = (s, c)$

$\textsc{Ver}$

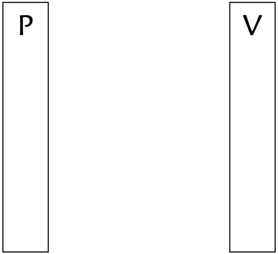
$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

## Schnorr Signatures (via the Fiat-Shamir Transform)

**Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$**

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.

$x \xleftarrow{\$} \mathbb{Z}_q$




**Signing and Verifying**

SIGN

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

VER

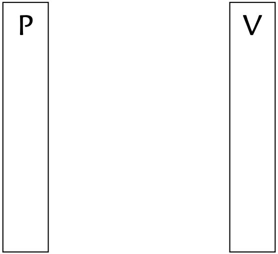
$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

## Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.

$x \xleftarrow{\$} \mathbb{Z}_q$

$y = g^x$




### Signing and Verifying

**Sign**

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

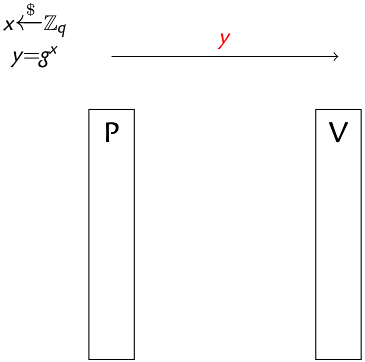
**Ver**

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

## Schnorr Signatures (via the Fiat-Shamir Transform)

**Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$**

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




**Signing and Verifying**

**Sign**

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

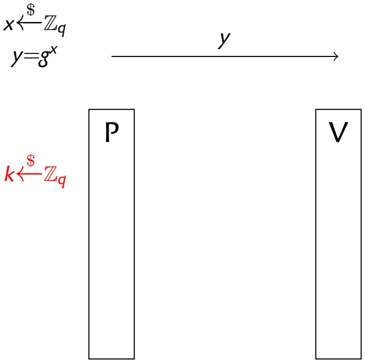
**Ver**

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

## Schnorr Signatures (via the Fiat-Shamir Transform)

### Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




### Signing and Verifying

Sign

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

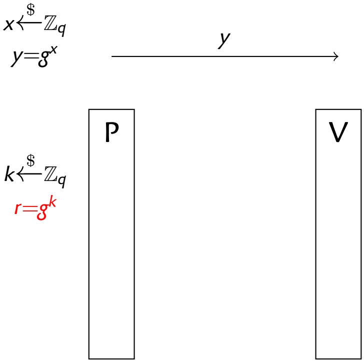
Ver

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

## Schnorr Signatures (via the Fiat-Shamir Transform)

### Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




### Signing and Verifying

**Sign**

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

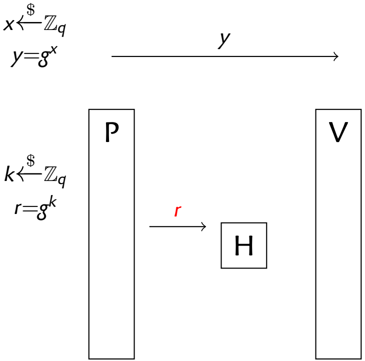
**Ver**

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

# Schnorr Signatures (via the Fiat-Shamir Transform)

## Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




## Signing and Verifying

SIGN

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \mod q$
$P$ sends $\sigma = (s, c)$

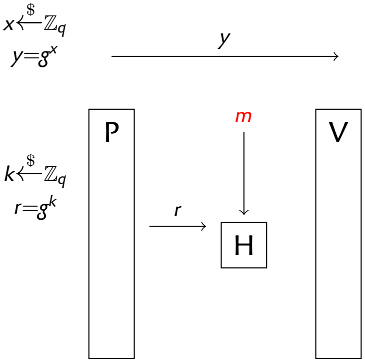
VER

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

## Schnorr Signatures (via the Fiat-Shamir Transform)

**Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$**

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




**Signing and Verifying**

SIGN

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

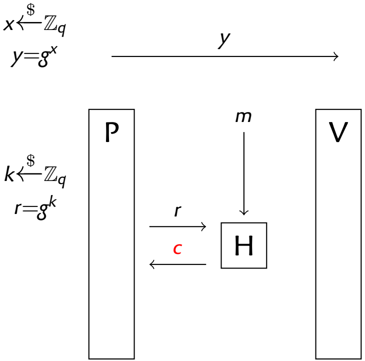
VER

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

## Schnorr Signatures (via the Fiat-Shamir Transform)

### Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




### Signing and Verifying

**SIGN**

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

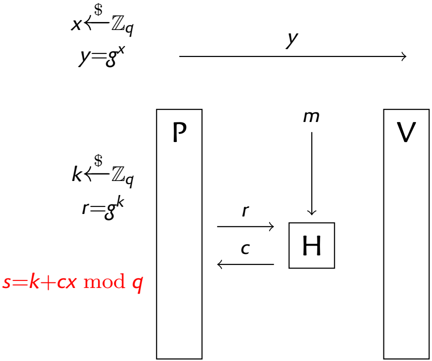
**VER**

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

## Schnorr Signatures (via the Fiat-Shamir Transform)

**Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$**

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




### Signing and Verifying

Sign

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$

$P$ computes $c = H(m, r)$

$P$ computes $s = k + cx \bmod q$

$P$ sends $\sigma = (s, c)$

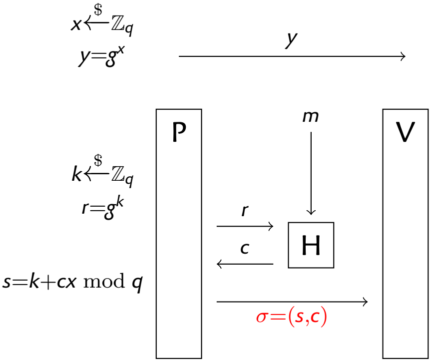
Ver

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

# Schnorr Signatures (via the Fiat-Shamir Transform)

## Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




## Signing and Verifying

**Sign**

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

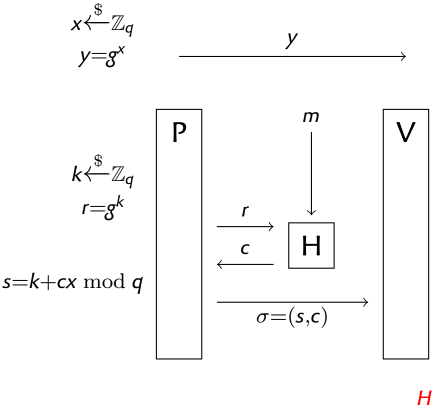
**Ver**

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

# Schnorr Signatures (via the Fiat-Shamir Transform)

**Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$**

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




**Signing and Verifying**

$\textsc{Sign}$

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \mod q$
$P$ sends $\sigma = (s, c)$

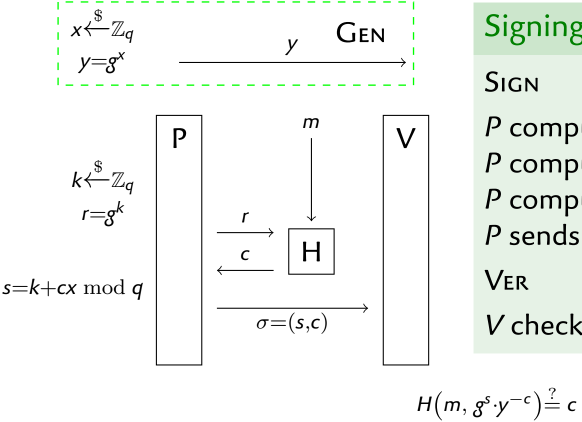
$\textsc{Ver}$

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

# Schnorr Signatures (via the Fiat-Shamir Transform)

## Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




## Signing and Verifying

**Sign**

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

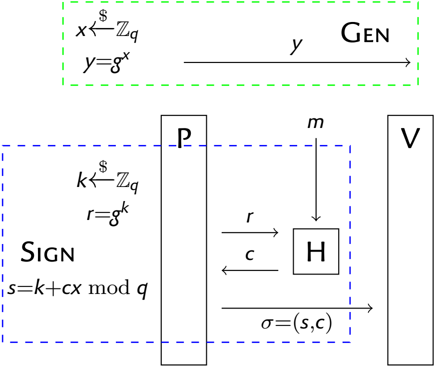
**Ver**

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

# Schnorr Signatures (via the Fiat-Shamir Transform)

**Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$**

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




$$H\left(m, g^s \cdot y^{-c}\right) \stackrel{?}{=} c$$

## Signing and Verifying

**Sign**

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

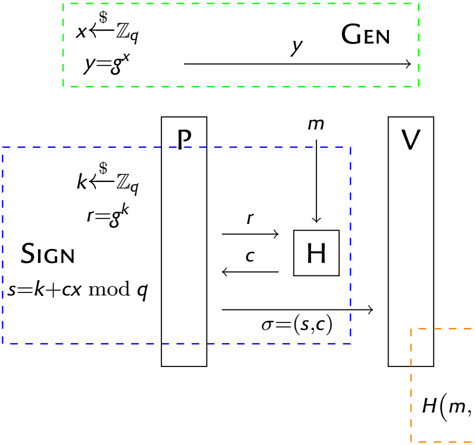
**Ver**

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

## Schnorr Signatures (via the Fiat-Shamir Transform)

### Introduce a hash function $H : \{0,1\}^\star \mapsto \mathbb{Z}_q$

Schnorr's signature scheme $\Sigma_H$ is a tuple of probabilistic algorithms $\Sigma_H = (\textsc{Gen}, \textsc{Sign}, \textsc{Ver})$ defined as follows.




### Signing and Verifying

**Sign**

$P$ computes $r = g^k$ where $k \xleftarrow{\$} \mathbb{Z}_q$
$P$ computes $c = H(m, r)$
$P$ computes $s = k + cx \bmod q$
$P$ sends $\sigma = (s, c)$

**Ver**

$V$ checks if $H\left(m, g^s \cdot y^{-c}\right) = c$

# Digression: Primality Certificates
## 1975

Claus Peter Schnorr
(1943–)

# Digital Signature Algorithm (DSA)

- The Digital Signature Algorithm (DSA) is a United States Federal Government standard or FIPS for digital signatures.
- It was proposed by the National Institute of Standards and Technology (NIST) in **August 1991** for use in their Digital Signature Standard (DSS), specified in FIPS 186, adopted in **1993**.
- DSA makes use of a cryptographic hash function $\mathcal{H}$.
- 2024: ECDSA with $\mathcal{H}$ := SHA256 is widespread

## Digital Signature Algorithm (DSA)

**Textbook ElGamal signature scheme (1985)**

Public parameters. A $k$-bit prime $p$ and a generator $g$ of $\mathbb{Z}_p^\times$

Key generation. The secret key is $x \xleftarrow{\$} \mathbb{Z}_{p-1}$
The public key is $y = g^x \mod p$

Signature. To sign a message $m \in \mathbb{Z}_{p-1}$, generate $(r, s)$ s.t.

$$g^m = y^r r^s \mod p$$

as follows: $k \xleftarrow{\$} \mathbb{Z}_{p-1}^\times$, $r \leftarrow g^k \mod p$ and

$$s \leftarrow (m - xr) \cdot k^{-1} \mod p - 1$$

Output $(r, s)$

Verification. Verify that $1 < r < p$ and $g^m \stackrel{?}{=} y^r r^s \mod p$

# Digital Signature Algorithm (DSA)

**Hashed ElGamal signature scheme**

Public parameters. A $k$-bit prime $p$ and a generator $g$ of $\mathbb{Z}_p^\times$

Key generation. The secret key is $x \xleftarrow{\$} \mathbb{Z}_{p-1}$
The public key is $y = g^x \mod p$

Signature. To sign a message $m \in \{0,1\}^*$, generate $(r, s)$ s.t.

$$g^{H(m)} = y^r r^s \mod p$$

as follows: $k \xleftarrow{\$} \mathbb{Z}_{p-1}^\times$, $r \leftarrow g^k \mod p$ and

$$ks \leftarrow (H(m) - xr) \cdot k^{-1} \mod p - 1$$

Output $(r, s)$

Verification. Verify that $1 < r < p$ and $g^{H(m)} \stackrel{?}{=} y^r r^s \mod p$

## Digital Signature Algorithm (DSA)

**Hashed ElGamal signature scheme with Schnorr's trick**

Public parameters. A $k$-bit prime $p$ and a generator $g \in \mathbb{Z}_p^\times$ of prime order $q$

Key generation. The secret key is $x \xleftarrow{\$} \mathbb{Z}_q$
The public key is $y = g^x \mod p$

Signature. To sign a message $m \in\in \{0,1\}^*$, generate $(r,s)$ s.t.

$$g^{H(m)} = y^r r^s \mod p$$

as follows: $k \xleftarrow{\$} \mathbb{Z}_q^\times$, $r \leftarrow g^k \mod p$ and

$$s \leftarrow (H(m) - xr) \cdot k^{-1} \mod q$$

Output $(r,s)$

Verification. Verify that $1 < r < q$ and $g^{H(m)} \stackrel{?}{=} y^r r^s \mod p$

# Digital Signature Algorithm (DSA)

## Full DSA

Public parameters. A $k$-bit prime $p$ and a generator $g \in \mathbb{Z}_p^\times$ of prime order $q$

Key generation. The secret key is $x \xleftarrow{\$} \mathbb{Z}_q$
The public key is $y = g^x \bmod p$

Signature. To sign a message $m \in \mathbb{Z}_{p-1}$, generate $(r, s)$ s.t.

$$g^{H(m)} = y^r r^s \bmod p$$

as follows: $k \xleftarrow{\$} \mathbb{Z}_q^\times$, $r \leftarrow \left(g^k \mod p\right) \bmod q$ and

$$s \leftarrow (H(m) + xr) \cdot k^{-1} \mod q$$

Output $(r, s)$

Verification. Verify that $1 < r < q$, compute $w \leftarrow s^{-1} \mod q$,
$u_1 = \mathcal{H}(m) \cdot w \bmod q$, $u_2 \leftarrow r \cdot w \bmod q$,
Check whether $(g^{u_1} y^{u_2} \bmod p) \bmod q \stackrel{?}{=} r$

Joseph-Louis Lagrange
(1736–1813)

## Theorem (Lagrange)

*Let $G$ be a finite group and $H \subseteq G$ a subgroup of $G$. Then $|H|$ divides $|G|$.*

## Proof.

- Let $x, y \in G$
- Say that $x \sim y$ iff $\exists h \in H$ (the subgroup) such that $x = yh$
- $\sim$ is an equivalence relation (easy)
- The equivalence class of $x$ is $xH$
- $xH$ has cardinality $|H|$
  - Multiplication by $x$ is a bijection in $G$
- Write $[G : H]$ the number of equivalence classes
  - Also known as the "*index* of $H$ in $G$"
- The equivalence classes form a partition of $G$
- Therefore $|G| = [G : H] \times |H|$

$\square$

# Interesting Consequence

### Corollary

*Let $\mathbb{G}$ be a finite group and $g \in \mathbb{G}$.*
*Then the order of $g$ divides the order of $\mathbb{G}$.*

### Proof.

$\langle x \rangle$ is a subgroup of $\mathbb{G}$. Apply Lagrange's theorem. $\square$

## Generators in $\mathbb{Z}_p^\times$

Let $q$ denote the order of $g$ modulo $p$

- $\mathbb{Z}_p^\times$ has order $p-1$
  - Notice that $p-1$ is even
  - $\{-1, 1\}$ is indeed a subgroup of order 2
- Therefore (Lagrange's theorem) $q$ divides $p-1$
  - $\rightsquigarrow$ *Considerably restricts* the possible values of $q$
- $q$ has a large prime factor $\Rightarrow p-1$ has a large prime factor
- $\mathbb{Z}_p^\times$ contains elements of order $p-1$
  - *Non-trivial theorem* (no proof given here)
  - This means that $\mathbb{Z}_p^\times$ is cyclic
  - An element of order $p-1$ is called a **primitive root** mod $p$

## Checking the Order of a Generator

### Problem

- Someone "promises" you that $g$ has order $q$ modulo $p$
- Can you verify that it is true?

### Validation?

- Check that $q$ divides $p - 1$
- Check that $g \neq 1$
- Check that $g^q = 1$ (necessary, **not sufficient**)
  - This proves that the actual order of $g$ divides $q$
  - It could be smaller than $q$
- Special case: the previous test is sufficient if $q$ **is prime**,

## Checking the Order of a Generator

### Problem

- Someone “promises” you that $g$ has order $q$ modulo $p$
- $q$ is **not prime** (relevant case: primitive roots)

### Validation?

- Let $\ell$ denote the actual order of $g$
- Check that $g^q = 1$ (necessary, **not sufficient**)
  - This proves that $\ell$ divides $q$
  - Write $q = \ell r$
- Suppose $\ell < q$ ($r \neq 1$)
  - Let $f$ be a prime factor of $r$ (and thus of $q$)
  - Then $g^{\frac{q}{f}} = g^{\frac{q}{r}\frac{r}{f}} = g^{\ell\frac{r}{f}} = 1^{\frac{r}{f}} = 1$
- Contrapositive:
  - $g^{\frac{q}{f}} \neq 1$ for each prime factor $f$ of $q \Longrightarrow g$ has order $q$

This procedure requires knowledge of the factorization of $q$

# Application: the “Oakley Groups” (RFC 2412 and 3526)

## Standardized Groups for the Masses

$$p = 2^{2048} - 2^{1984} - 1 + 2^{64} \times \left([2^{1918}\pi] + 124476\right)$$
$$g = 2$$

Claim : $g$ has order $p - 1$ modulo $p$

**Proof.**

- Let $q$ denote the order of $g$
- $\ell = (p-1)/2$ is also prime
  - $p$ is a *Sophie Germain* prime or a *safe* prime
- Therefore $q \in \{2, \ell, 2\ell\}$
- $g^2 \neq 1$ and $g^\ell \neq 1$, therefore $g$ has order $p - 1$

$\square$

Conclusion: $\mathbb{Z}_p^\times = \langle 2 \rangle$

## Creating Generators of Prime Order in $\mathbb{Z}_p^\times$ — Schnorr's Trick

### Procedure

1. Choose a 256-bit prime $q$
2. Pick a random 1792-bit integer $k$
3. Set $p = 1 + kq$
4. If $p$ is not prime, go back to 2.
5. Pick a random $x$ modulo $p$
6. Set $g \leftarrow x^k$
7. If $g = 1$, go back to 5.
8. $g$ has (prime) order $q$ modulo $p$

Proof.

- $g^q = x^{p-1} = 1$
  - By Fermat's little theorem
- Therefore, if $g \neq 1$, then $g$ has order $q$
  - cf. previous slides (easy case: $q$ is prime) □

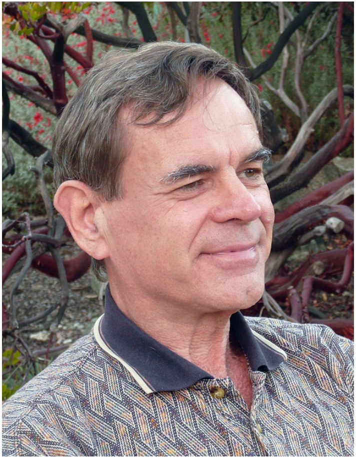
## Digression: Primality Certificates
### 1975

**If $g$ has order $n-1$ modulo $n$, then $n$ is prime**

- $\langle g \rangle \subseteq \mathbb{Z}_n^\times$
- $g$ has order $n-1$, *therefore* $|\mathbb{Z}_n^\times| = n-1$
- All integers except zero are invertible modulo $n$
- $n$ does not have any non-trivial divisor
- $n$ is prime

- providing $g$ of order $n-1$ proves that $n$ is prime
- Checking the order of $g$ requires the factorization of $n-1$
- Certificate of $n$ =
  1. $g$
  2. Factorization of $n-1$
  3. Certificates of the prime factors (recursively)
- Conclusion: PRIMES $\in$ NP

# Digression: Primality Certificates
## 1975



Vaughan Pratt
(1944–)

# DDH Can be Easier than CDH

Let $g$ be a **primitive root** modulo $p$

- **DLOG** and **CDH** are (presumably) hard in $\mathbb{Z}_p^\times$
- But **DDH** is easy in $\mathbb{Z}_p^\times$!!!
- Argument given around 1800

Leonhard Euler
1707–1783

Adrien-Marie Legendre
1752–1833

## Quadratic Residuosity

### Definition

Quadratic Residue $x \in \mathbb{Z}_p^\times$ is a **quadratic residue** $\Leftrightarrow$ $x$ is a square $(\exists y.\, x = y^2)$

- $x$ and $-x$ have the same square

$\rightsquigarrow$ $(p-1)/2$ quadratic residues

- "Fun" : $25^2 = 5 \mod 31$

### Important because...

**It is easy to test if $x \in \mathbb{Z}_p$ is a quadratic residue**

## Proposition

*Let $g$ be a primitive root modulo $p > 2$. Then*

$$g^x \text{ is a quadratic residue} \iff x \equiv 0 \bmod 2$$

## Proof.

$\Leftarrow$ Trivial. $x \equiv 0 \bmod 2 \Rightarrow \exists y.x = 2y \Rightarrow g^x = g^{2y} = (g^y)^2$

$\Rightarrow$ Suppose that $g^x = \alpha^2$

- $g$ is a primitive root: $\exists y.\alpha = g^y$

$\rightsquigarrow$ $g^x = \alpha^2 = (g^y)^2 = g^{2y}$

- Therefore (lemma from last week)

$$x \equiv 2y \bmod p - 1 \quad \Rightarrow \quad \exists k.x = 2y + k(p-1)$$

- $p$ is odd $\rightsquigarrow p - 1 = 2\ell$, so $x = 2(y + k\ell)$
- $x$ is even

$\square$

## One-Way Functions?

### Exponentiation mod $p$ : $x \mapsto g^x$

- I claimed that it is one-way...
  - $\mathcal{A}$ does not recover $x$ from $F(x)$

$\xleftarrow{F(x)}$

$x$

Adversary $\mathcal{A}$

Challenger

$\rightarrow x$

## One-Way Functions?

### Exponentiation mod $p$ : $x \mapsto g^x$

- I claimed that it is one-way...
  - $\mathcal{A}$ does not recover $x$ from $F(x)$
- Could $\mathcal{A}$ recover **one bit** $P(x)$ of information about $x$?

$\xleftarrow{F(x)}$

$x$

Adversary $\mathcal{A}$

Challenger

$\longrightarrow P(x)$

## Legendre Symbol and Euler's Criterion

### Definition (Legendre Symbol)

Let $p$ be an odd prime number.

$$\left(\frac{a}{p}\right) \stackrel{def}{=} \begin{cases} 1 & \text{if } a \text{ is a quadratic residue mod } p \\ 0 & \text{if } a = 0 \\ -1 & \text{if } a \text{ is a not quadratic residue mod } p \end{cases}$$

- The Legendre symbol is just a weird notation for this specific function

### Theorem: Euler's Criterion

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \mod p$$

# Weak Bits of the Discrete Logarithm

**Exponentiation mod $p$ : $x \mapsto g^x$**

With $g$ a primitive root modulo $p$

$\xleftarrow{\quad g^x \quad}$

$x$

Adversary $\mathcal{A}$

Challenger

$x \mod 2$

(Euler's criterion)

## Euler's Criterion: $p > 2$ prime $\Rightarrow \left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \mod p$

### Proof.

Let's work inside the finite field $\mathbb{Z}_p$.

$$P(X) = X^{p-1} - 1 = \left(X^{\frac{p-1}{2}}\right)^2 - 1 = \underbrace{\left(X^{\frac{p-1}{2}} - 1\right)}_{P_1(X)} \underbrace{\left(X^{\frac{p-1}{2}} + 1\right)}_{P_{-1}(X)}$$

1. $\alpha$ is a QR $\Longrightarrow \alpha^{\frac{p-1}{2}} \equiv 1 \mod p$
   Let $\alpha = \beta^2$ be a quadratic residue. Then
   $$P_1(\alpha) = P_1\left(\beta^2\right) = \left(\beta^2\right)^{\frac{p-1}{2}} - 1 = \beta^{p-1} - 1 = 0$$
   (last step by Fermat's little theorem — everything mod $p$)
2. $\alpha$ is not a QR $\Longrightarrow P_1(\alpha) \neq 0$
   Note that $P_1(0) = -1$, so that $P_1(X) \neq 0$
   $P_1(X)$ vanishes over the $(p-1)/2$ quadratic residues
   $\deg P_1 = (p-1)/2 \rightsquigarrow P_1$ cannot have any more roots

## Euler's Criterion: $\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \mod p$

**Proof.**

$$P(X) = X^{p-1} - 1 = \left(X^{\frac{p-1}{2}}\right)^2 - 1 = \underbrace{\left(X^{\frac{p-1}{2}} - 1\right)}_{P_1(X)} \underbrace{\left(X^{\frac{p-1}{2}} + 1\right)}_{P_{-1}(X)}$$

1. $\alpha$ is a QR $\implies \alpha^{\frac{p-1}{2}} = 1$
2. $\alpha$ is not a QR $\implies P_1(\alpha) \neq 0$
3. $\alpha$ is not a QR $\implies \alpha^{\frac{p-1}{2}} = -1$
   - Fermat's little theorem $\Rightarrow P(\alpha) = 0$
   - $P_1(\alpha) \neq 0 \implies P_{-1}(\alpha) = 0$
   - (everything mod $p$ again)

$\square$