Intro to IT Security

CS306C—Fall 2022

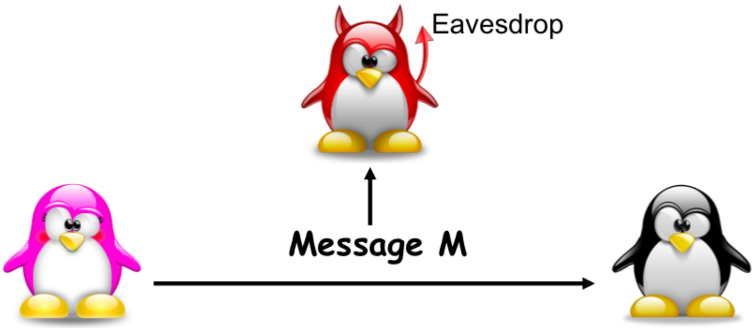
Prof. Antonio R. Nicolosi

Antonio.Nicolosi@stevens.edu

STEVENS INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY

Asymmetric Encryption Schemes

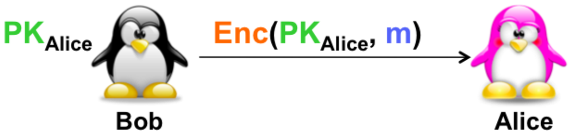
Crypto Requirements: Data Secrecy



- Protect against unauthorized disclosure of the msg
  - If **A** sends a msg to **B**, no one else should understand its content

Asymmetric Encryption

- Defined by three algorithms:
  - $\mathsf{Gen}(\lambda) \rightarrow (SK, PK)$ outputs secret key SK and public key PK
  - $\mathsf{Enc}(PK, m) \rightarrow c$ encrypt m using public key PK
  - $\mathsf{Dec}(SK, c) \rightarrow m$ or $\perp$ decrypt c using SK



Asymmetric Encryption

- Bob wants to send a secret message m to Alice
- Alice creates a (encryption/decryption) key pair (pk_A, sk_A)
- Alice publishes the encryption key pk_A and keeps secret sk_A
- Bob uses Alice's encryption key pk_A to generate an encryption of m
- Alice recovers the secret message m using her decryption key sk_A

ElGamal Encryption Scheme

Let p be a prime and consider the finite field of order p.

Let $\mathbb{G} \subset \mathbb{Z}_p$ be a cyclic group of order q ($|q| = n$), $q|(p-1)$. Let g be a generator.

- $KG(1^n, \mathbb{G}, q, g)$: choose random $x \leftarrow \mathbb{Z}_q$ and compute $h := g^x \mod p$

$$pk \stackrel{def}{=} \langle \mathbb{G}, q, g, h \rangle \qquad sk \stackrel{def}{=} x$$

- $Enc_{pk}(m)$: choose random $r \leftarrow \mathbb{Z}_q$ and outputs the ciphertext

$$c := (g^r \mod p, h^r \cdot m \mod p)$$

- $Dec_{sk}(c = (c_1, c_2))$: outputs $m := c_2 \cdot (c_1^x)^{-1} \mod p$

$$\begin{aligned} c_2 \cdot (c_1^x)^{-1} \mod p &= h^r \cdot m \cdot ((g^r)^x)^{-1} \mod p \\ &= (g^x)^r \cdot m \cdot ((g^r)^x)^{-1} \mod p \\ &= m \mod p \end{aligned}$$

ElGamal Encryption Scheme

Let p be a prime and consider the finite field of order p.

Let $\mathbb{G} \subset \mathbb{Z}_p$ be a cyclic group of order q ($|q| = n$), $q|(p-1)$. Let g be a generator.

- $KG(1^n, \mathbb{G}, q, g)$: choose random $x \leftarrow \mathbb{Z}_q$ and compute $h := g^x \mod p$

$$pk \stackrel{def}{=} \langle \mathbb{G}, q, g, h \rangle \qquad sk \stackrel{def}{=} x$$

- $Enc_{pk}(m)$: choose random $r \leftarrow \mathbb{Z}_q$ and outputs the ciphertext

$$c := (g^r \mod p, h^r \cdot m \mod p)$$

- $Dec_{sk}(c = (c_1, c_2))$: outputs $m := c_2 \cdot (c_1^x)^{-1} \mod p$

$$\begin{aligned} c_2 \cdot (c_1^x)^{-1} \mod p &= h^r \cdot m \cdot ((g^r)^x)^{-1} \mod p \\ &= (g^x)^r \cdot m \cdot ((g^r)^x)^{-1} \mod p \\ &= m \mod p \end{aligned}$$

ElGamal Encryption Scheme

Let p be a prime and consider the finite field of order p.

Let $\mathbb{G} \subset \mathbb{Z}_p$ be a cyclic group of order q ($|q| = n$), $q|(p-1)$. Let g be a generator.

- $KG(1^n, \mathbb{G}, q, g)$: choose random $x \leftarrow \mathbb{Z}_q$ and compute $h := g^x \mod p$

$$pk \stackrel{def}{=} \langle \mathbb{G}, q, g, h \rangle \qquad sk \stackrel{def}{=} x$$

- $Enc_{pk}(m)$: choose random $r \leftarrow \mathbb{Z}_q$ and outputs the ciphertext

$$c := (g^r \mod p, h^r \cdot m \mod p)$$

- $Dec_{sk}(c = (c_1, c_2))$: outputs $m := c_2 \cdot (c_1^x)^{-1} \mod p$

$$\begin{aligned} c_2 \cdot (c_1^x)^{-1} \mod p &= h^r \cdot m \cdot ((g^r)^x)^{-1} \mod p \\ &= (g^x)^r \cdot m \cdot ((g^r)^x)^{-1} \mod p \\ &= m \mod p \end{aligned}$$

ElGamal Encryption Scheme

Let p be a prime and consider the finite field of order p.

Let $\mathbb{G} \subset \mathbb{Z}_p$ be a cyclic group of order q ($|q| = n$), $q|(p-1)$. Let g be a generator.

- $KG(1^n, \mathbb{G}, q, g)$: choose random $x \leftarrow \mathbb{Z}_q$ and compute $h := g^x \mod p$
$$pk \stackrel{def}{=} \langle \mathbb{G}, q, g, h \rangle \qquad sk \stackrel{def}{=} x$$

- $Enc_{pk}(m)$: choose random $r \leftarrow \mathbb{Z}_q$ and outputs the ciphertext
$$c := (g^r \mod p, h^r \cdot m \mod p)$$

- $Dec_{sk}(c = (c_1, c_2))$: outputs $m := c_2 \cdot (c_1^x)^{-1} \mod p$
$$\begin{aligned} c_2 \cdot (c_1^x)^{-1} \mod p &= h^r \cdot m \cdot ((g^r)^x)^{-1} \mod p \\ &= (g^x)^r \cdot m \cdot ((g^r)^x)^{-1} \mod p \\ &= m \mod p \end{aligned}$$

Security of the ElGamal Encryption Scheme

Theorem. *If the DDH problem is hard relative to* $\mathbb{G}$*, then the ElGamal encryption scheme has indistinguishable encryption under chosen plaintext attack.*

Recall...

- Euler Theorem
  - $x^{\phi(n)} \equiv 1 \mod n$
- Fermat Theorem
  - p prime
  - $x^{p-1} \equiv 1 \mod p$ (in $\mathbb{Z}_p$)
  - equivalently $x^p \equiv x \mod p$ (in $\mathbb{Z}_p*$)

Basic RSA Algorithm

(RSA = Rivest-Shamir-Adleman)

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
$$gcd(e, \phi(n)) = 1$$
    (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Let $pk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $fwd_{pk}(m) = m^e \mod n$
- $bwd_{sk}(c) = c^d \mod n$
- Note: $(m^e)^d \equiv m \mod n$

  Since $ed = 1 + h\phi(n)$, we have
$$\begin{aligned} m^{ed} &\equiv m^{1+h\phi(n)} \mod n \\ &\equiv m \cdot (m^{\phi(n)})^h \mod n \\ &\equiv m \cdot 1^h \mod n \\ &\equiv m \mod n \end{aligned}$$

Basic RSA Algorithm

(RSA = Rivest-Shamir-Adleman)

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
    $$gcd(e, \phi(n)) = 1$$
    (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Let $pk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $fwd_{pk}(m) = m^e \mod n$
- $bwd_{sk}(c) = c^d \mod n$
- Note: $(m^e)^d \equiv m \mod n$

  Since $ed = 1 + h\phi(n)$, we have
  $$\begin{aligned} m^{ed} &\equiv m^{1+h\phi(n)} \mod n \\ &\equiv m \cdot (m^{\phi(n)})^h \mod n \\ &\equiv m \cdot 1^h \mod n \\ &\equiv m \mod n \end{aligned}$$

Basic RSA Algorithm

(RSA = Rivest-Shamir-Adleman)

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
  $$gcd(e, \phi(n)) = 1$$
  (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Let $pk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $fwd_{pk}(m) = m^e \mod n$
- $bwd_{sk}(c) = c^d \mod n$
- Note: $(m^e)^d \equiv m \mod n$

  Since $ed = 1 + h\phi(n)$, we have
  $$\begin{aligned} m^{ed} &\equiv m^{1+h\phi(n)} \mod n \\ &\equiv m \cdot (m^{\phi(n)})^h \mod n \\ &\equiv m \cdot 1^h \mod n \\ &\equiv m \mod n \end{aligned}$$

Basic RSA Algorithm

(RSA = Rivest-Shamir-Adleman)

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
    $$gcd(e, \phi(n)) = 1$$
    (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Let $pk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $fwd_{pk}(m) = m^e \mod n$
- $bwd_{sk}(c) = c^d \mod n$
- Note: $(m^e)^d \equiv m \mod n$

  Since $ed = 1 + h\phi(n)$, we have
  $$\begin{aligned} m^{ed} &\equiv m^{1+h\phi(n)} \mod n \\ &\equiv m \cdot (m^{\phi(n)})^h \mod n \\ &\equiv m \cdot 1^h \mod n \\ &\equiv m \mod n \end{aligned}$$

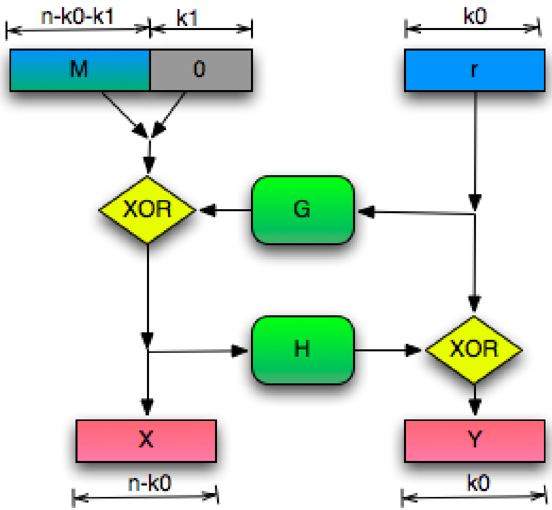
Basic “Textbook” RSA is not a Secure Encryption Scheme

- The RSA algorithm is deterministic
  - Cannot work for encryption
  - Proper terminology is *trapdoor permutation*
- Several attacks have been shown in the past

$\Rightarrow$ Need a padding scheme: OAEP

Optimal Asymmetric Encryption Padding (OAEP)

- n is the number of bits in the RSA modulus.
- k_0 and k_1 are integers fixed by the protocol.
- m is the plaintext message, a $(n - k_0 - k_1)$-bit string
- G and H are typically some cryptographic hash functions fixed by the protocol.



Optimal Asymmetric Encryption Padding (OAEP)

To encode:

1. messages are padded with k_1 zeros to be $n-k_0$ bits in length.
2. r is a random k_0 bit string
3. G expands the k_0 bits of r to $n-k_0$ bits.
4. $X = m00..0 \oplus G(r)$
5. H reduces the $n-k_0$ bits of X to k_0 bits.
6. $Y = r \oplus H(X)$
7. The output is $X||Y$ where X is shown in the diagram as the leftmost block and Y as the rightmost block.

Optimal Asymmetric Encryption Padding (OAEP)

To decode:

1. recover the random string as $r = Y \oplus H(X)$
2. recover the message as $m00..0 = X \oplus G(r)$

OAEP-RSA Encryption Scheme

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q;
  - Let $\phi(n) = (p-1)(q-1)$.
  - Choose e such that
    $$gcd(e, \phi(n)) = 1$$
    (most random e will work);
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.);
  - Let $pk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $Enc_{pk}(m, r) = [\mathrm{OAEP}(m, r)]^e \mod n = [X||Y]^e \mod n$
- $Dec_{sk}(c) =$
  1. $(X||Y) \leftarrow c^d \mod n$
  2. $m \leftarrow \mathrm{OAEP}^{-}1(X||Y)$

OAEP-RSA Encryption Scheme

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q;
  - Let $\phi(n) = (p-1)(q-1)$.
  - Choose e such that
    $$gcd(e, \phi(n)) = 1$$
    (most random e will work);
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.);
  - Let $pk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $Enc_{pk}(m, r) = [\text{OAEP}(m, r)]^e \mod n = [X||Y]^e \mod n$
- $Dec_{sk}(c) =$
  1. $(X||Y) \leftarrow c^d \mod n$
  2. $m \leftarrow \text{OAEP}^{-}1(X||Y)$

OAEP-RSA Encryption Scheme

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q;
  - Let $\phi(n) = (p-1)(q-1)$.
  - Choose e such that
  $$gcd(e, \phi(n)) = 1$$
  (most random e will work);
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.);
  - Let $pk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $Enc_{pk}(m, r) = [\text{OAEP}(m, r)]^e \mod n = [X||Y]^e \mod n$
- $Dec_{sk}(c) =$
  1. $(X||Y) \leftarrow c^d \mod n$
  2. $m \leftarrow \text{OAEP}^{-}1(X||Y)$

OAEP-RSA Encryption Scheme

Specifically:

- $Enc_{pk}(m, r)$:

  1. $X \stackrel{def}{=} m0\ldots0 \oplus G(r) \mod n$
  2. $Y \stackrel{def}{=} r \oplus H(X)$
  3. $c \stackrel{def}{=} (X||Y)^e \mod n$

- $Dec_{sk}(c)$:

  1. $(X||Y) \leftarrow c^d \mod n$
  2. $r = Y \oplus H(X)$
  3. $m0\ldots0 = X \oplus G(r)$

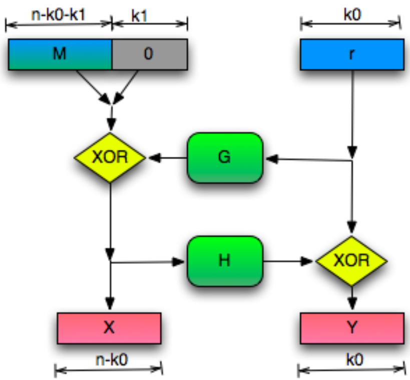
OAEP-RSA Encryption Scheme

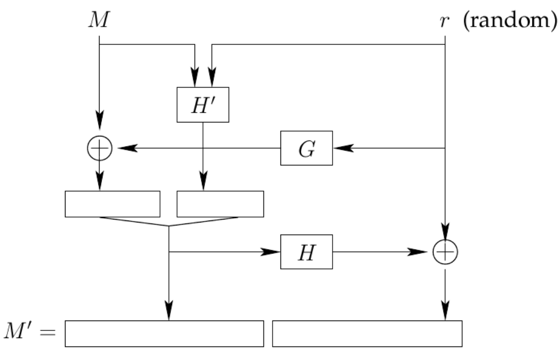
Specifically:

- $Enc_{pk}(m, r)$:
  1. $X \stackrel{def}{=} m0\ldots0 \oplus G(r) \mod n$
  2. $Y \stackrel{def}{=} r \oplus H(X)$
  3. $c \stackrel{def}{=} (X||Y)^e \mod n$
- $Dec_{sk}(c)$:
  1. $(X||Y) \leftarrow c^d \mod n$
  2. $r = Y \oplus H(X)$
  3. $m0\ldots0 = X \oplus G(r)$

OAEP+

- OAEP-RSA is CCA-secure
  - But OAEP does not work with all trapdoor permutations...
- OAEP+ works for every family of trapdoor permutations





Intro to IT Security

CS306C—Fall 2022

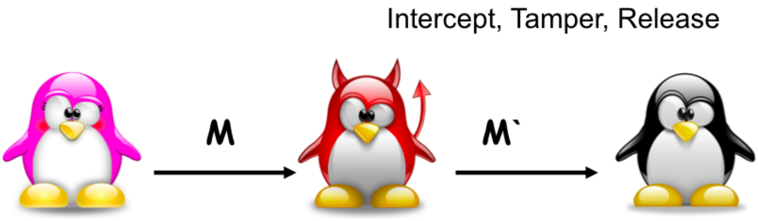
Prof. Antonio R. Nicolosi

Antonio.Nicolosi@stevens.edu

STEVENS INSTITUTE of TECHNOLOGY
THE INNOVATION UNIVERSITY

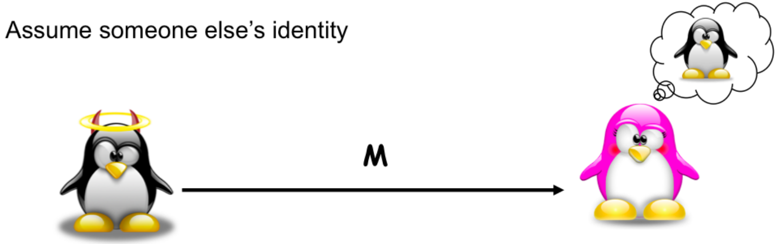
Digital Signatures

Crypto Requirements: Data Integrity



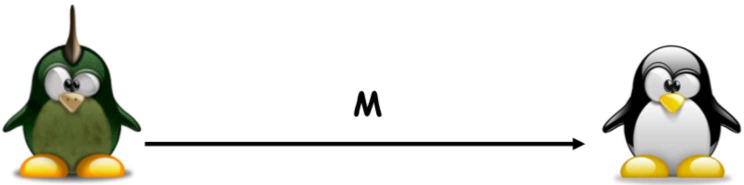
- The receiver should be able to check whether the msg was modified during transmission
  - No one should be able to tamper with the msg, without the recipient noticing the alteration

Crypto Requirements: Data Origin



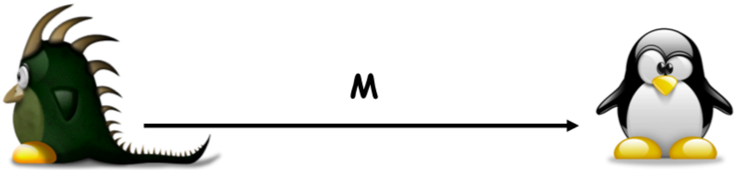
- The receiver of a msg should be able to verify its origin
  - No one else should be able to send a msg to **A** pretending to be **B**

Crypto Requirements: Non-Repudiation



- The sender should not be able to later deny responsibility for msgs sent

Crypto Requirements: Non-Repudiation



- The sender should not be able to later deny responsibility for msgs sent

Digital Signature: Scenario

- Alice wants to send Bob a message that he knows came from her
- With paper messages, she would sign her name
- What to do in the electronic setting?
- Alice would like a way to digitally sign documents that maintains the properties we usually expect from a signature:
  - Anyone who knows Alice should be able to verify a legitimate signature
  - No one should be able to forge a signature

Digital Signature *vs.* MAC

- Public verifiability
- Transferability
- Non repudiation

Digital Signature

- Defined by three algorithms:
  - $\mathsf{Gen}(\lambda) \rightarrow (SK, VK)$ outputs secret key SK and public key VK
  - $\mathsf{Sign}(SK, m) \rightarrow (m, \sigma)$ sign m using secret signing key SK
  - $\mathsf{Vrfy}(VK, m, \sigma) \rightarrow 1/0$ verify σ using m and VK

Systems with passwords: Examples

- Bad examples: PGP & ssh private keys
  - Stored in user's home directory, encrypted with password
  - Given encrypted private key, can guess password off-line
  - Cost of guess comparable to that of password `crypt`
- Good example: OpenBSD/FreeBSD `bcrypt`
  - Hashing password is exponential in cost parameter
  - Cost goes in hashed password along with salt

Network Passwords

- Many systems grant access through a password
- How to implement? Example:
  - Server stores user's password (or hash)
  - Client connects to server, sends username, password
  - Server compares password to stored version
  - Grants access if they match
- Is this a good approach?

Digital Signature: Correctness

$$\Pr[Ver_{vk}(m, Sign_{sk}(m)) = 1] = 1$$

where the probability is over $(sk, vk) \leftarrow KG(1^n)$, and the randomness used in the signing algorithm $Sign$

Digital Signature: Security

Given a signature scheme $\Pi = (KG, Sign, Ver)$, and an adversary $\mathcal{A}$, consider the following experiment.

Let $(sk, vk) = KG(1^n)$:

Attacker$(1^n, vk)$ Challenger

$$\xrightarrow{m_0,\ldots,m_t}$$

$$\xleftarrow{\sigma_0 = Sign_{sk}(m_0),\ldots,\sigma_t = Sign_{sk}(m_t)}$$

$$\xrightarrow{(m^*,\sigma^*)} \quad Ver_{vk}(m^*, \sigma^*)$$

The output of the experiment is 1 if $Ver_{vk}(m^*, \sigma^*) = 1$, and 0 otherwise.

Basic RSA Algorithm

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
    $$gcd(e, \phi(n)) = 1$$
    (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Let $vk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $Sign_{sk}(m) : \sigma = m^d \mod n$
- $Ver_{vk}(m, \sigma) : m \stackrel{?}{=} \sigma^e \mod n$
- Note: $(m^e)^d \equiv m \mod n$

  Since $ed \equiv 1 \mod \phi(n)$, we have
  $$\sigma^e \mod n = (m^d)^e \mod n = m \mod n$$

Basic RSA Algorithm

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
  $$gcd(e, \phi(n)) = 1$$
  (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Let $vk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $Sign_{sk}(m) : \sigma = m^d \mod n$
- $Ver_{vk}(m, \sigma) : m \stackrel{?}{=} \sigma^e \mod n$
- Note: $(m^e)^d \equiv m \mod n$

  Since $ed \equiv 1 \mod \phi(n)$, we have
  $$\sigma^e \mod n = (m^d)^e \mod n = m \mod n$$

Basic RSA Algorithm

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
$$gcd(e, \phi(n)) = 1$$
    (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Let $vk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $Sign_{sk}(m) : \sigma = m^d \mod n$
- $Ver_{vk}(m, \sigma) : m \stackrel{?}{=} \sigma^e \mod n$
- Note: $(m^e)^d \equiv m \mod n$

  Since $ed \equiv 1 \mod \phi(n)$, we have
$$\sigma^e \mod n = (m^d)^e \mod n = m \mod n$$

Basic RSA Algorithm

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
  $$gcd(e, \phi(n)) = 1$$
  (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Let $vk \stackrel{def}{=} (n, e)$ and $sk \stackrel{def}{=} (n, d)$.
- $Sign_{sk}(m) : \sigma = m^d \mod n$
- $Ver_{vk}(m, \sigma) : m \stackrel{?}{=} \sigma^e \mod n$
- Note: $(m^e)^d \equiv m \mod n$

  Since $ed \equiv 1 \mod \phi(n)$, we have
  $$\sigma^e \mod n = (m^d)^e \mod n = m \mod n$$

Basic “Textbook” RSA is not a Secure Signature Scheme

- Knowing only the public parameters, can create forgery on a random message
  - pick a random signature σ and set the message to be $\sigma^e \mod n$
- Easy to forge a signature on a chosen message, given two signatures of adversary choice:
  - given $\sigma_1 = m_1^d \mod n$ and $\sigma_2 = m_2^d \mod n$:

$$\sigma_1 * \sigma_2 = (m_1^d)(m_2^d) \mod n = (m_1 m_2)^d \mod n$$

- In general, given a signature σ for a message m, one can create a signature on a related message m^t
  - Note: $\sigma^t = (m^d)^t = (m^t)^d \mod n$

$\Rightarrow$ Need a padding scheme: Hashed-RSA

Hashed-RSA

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
    $$gcd(e, \phi(n)) = 1$$
    (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Select $H : \{0,1\}^* \to \mathbb{Z}_n^*$, collision-resistant
  - Let $vk \stackrel{def}{=} (n, e, H)$ and $sk \stackrel{def}{=} (n, d)$
- $Sign_{sk}(m) : \sigma = H(m)^d \mod n$
- $Ver_{sk}(m, \sigma) : \sigma^e \stackrel{?}{=} H(m) \mod n$

Hashed-RSA

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
    $$gcd(e, \phi(n)) = 1$$
    (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Select $H : \{0,1\}^* \to \mathbb{Z}_n^*$, collision-resistant
  - Let $vk \stackrel{def}{=} (n, e, H)$ and $sk \stackrel{def}{=} (n, d)$
- $Sign_{sk}(m) : \sigma = H(m)^d \mod n$
- $Ver_{sk}(m, \sigma) : \sigma^e \stackrel{?}{=} H(m) \mod n$

Hashed-RSA

- $KG(1^n)$:
  - Compute n as the product of two k-bit primes p and q
  - Let $\phi(n) = (p-1)(q-1)$
  - Choose e such that
    $$gcd(e, \phi(n)) = 1$$
    (most random e will work)
  - Compute $d \equiv e^{-1} \mod \phi(n)$ (extended Euclidean alg.)
  - Select $H : \{0,1\}^* \to \mathbb{Z}_n^*$, collision-resistant
  - Let $vk \stackrel{def}{=} (n, e, H)$ and $sk \stackrel{def}{=} (n, d)$
- $Sign_{sk}(m) : \sigma = H(m)^d \mod n$
- $Ver_{sk}(m, \sigma) : \sigma^e \stackrel{?}{=} H(m) \mod n$

Schnorr Signature Scheme

- KG: Let q be a prime and let $p = 2q + 1$. Let g be a generator of prime order q. Pick $x \xleftarrow{\$} \mathbb{Z}_q$ and set $y = g^x \mod p$.

$$sk \stackrel{def}{=} x \qquad vk \stackrel{def}{=} y$$

- $Sign_{sk}(m)$:
  1. $r \xleftarrow{\$} \mathbb{Z}_q$
  2. $a := g^r \mod p$
  3. $H : \{0,1\}^* \to \mathbb{Z}_q, \quad c := H(y||m||a)$
  4. $z := r + x \cdot c \mod q$
  5. $\sigma \stackrel{def}{=} (a, z)$
- $Ver_{vk}(m, \sigma)$:
  1. $c := H(y||m||a)$
  2. $g^z \mod p \stackrel{?}{=} a \cdot y^c \mod p$

Schnorr Signature Scheme

- *KG*: Let q be a prime and let $p = 2q+1$. Let g be a generator of prime order q. Pick $x \xleftarrow{\$} \mathbb{Z}_q$ and set $y = g^x \mod p$.

$$sk \stackrel{def}{=} x \qquad vk \stackrel{def}{=} y$$

- $Sign_{sk}(m)$:
  1. $r \xleftarrow{\$} \mathbb{Z}_q$
  2. $a := g^r \mod p$
  3. $H : \{0,1\}^* \rightarrow \mathbb{Z}_q, \quad c := H(y||m||a)$
  4. $z := r + x \cdot c \mod q$
  5. $\sigma \stackrel{def}{=} (a, z)$
- $Ver_{vk}(m, \sigma)$:
  1. $c := H(y||m||a)$
  2. $g^z \mod p \stackrel{?}{=} a \cdot y^c \mod p$

Schnorr Signature Scheme

- *KG*: Let q be a prime and let $p = 2q + 1$. Let g be a generator of prime order q. Pick $x \xleftarrow{\$} \mathbb{Z}_q$ and set $y = g^x \mod p$.

$$sk \stackrel{def}{=} x \qquad vk \stackrel{def}{=} y$$

- $Sign_{sk}(m)$:
  1. $r \xleftarrow{\$} \mathbb{Z}_q$
  2. $a := g^r \mod p$
  3. $H : \{0,1\}^* \rightarrow \mathbb{Z}_q, \quad c := H(y||m||a)$
  4. $z := r + x \cdot c \mod q$
  5. $\sigma \stackrel{def}{=} (a, z)$
- $Ver_{vk}(m, \sigma)$:
  1. $c := H(y||m||a)$
  2. $g^z \mod p \stackrel{?}{=} a \cdot y^c \mod p$

Schnorr Signature Scheme: Security

- Schnorr signature scheme is existantial unforgeable under adaptive chosen-message attacks, under the discrete log assumption.

Hash-then-Sign

- Say we have a secure signature scheme for short messages (e.g. hashed RSA, Schnorr)
- How can we use it to sign long messages?
  1. Hash the long message: $H(m)$
  2. Sign the digested message: $\sigma = Sign(H(m), sk)$
- But does this work?
  - If $H(m_1) = H(m_2)$, then signature on m_1 is valid for m_2, too!
  - If H is collision resistant, hard to find two messages with same digest