crypto attacks & defenses

RELOADED

JP Aumasson, Philipp Jovanovic

ringzerø

public-key crypto

Overview

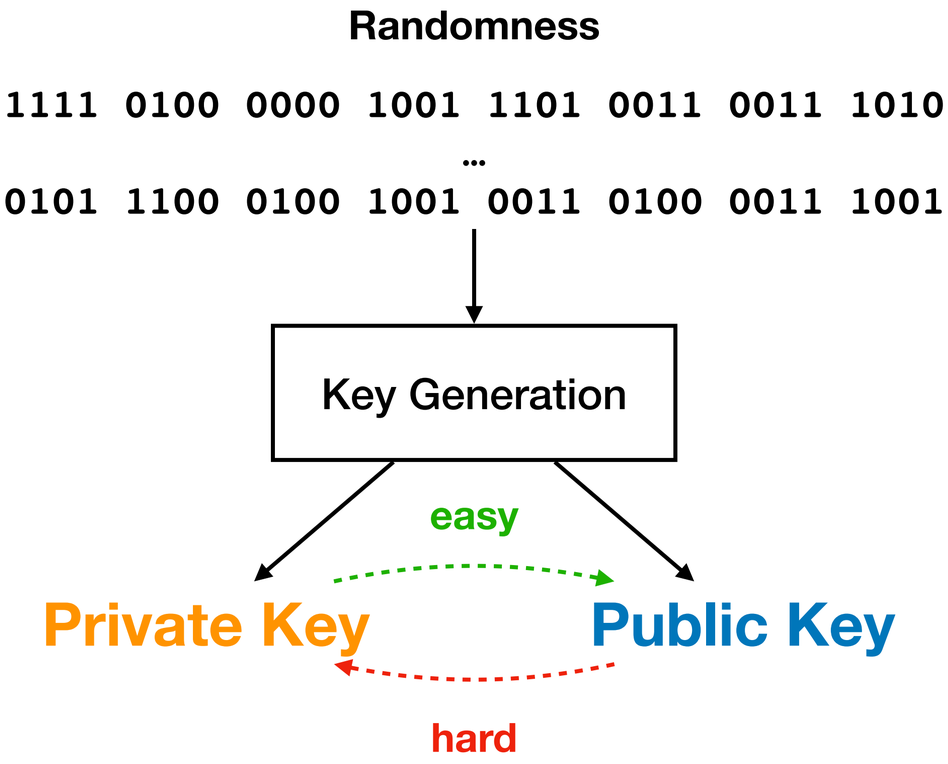
- Computationally hard problems
- Public key encryption
- Digital signatures
- Key exchange
- Hybrid encryption

Core Goals of Asymmetric Cryptography

- **Confidentiality:** Ensure that the secrecy of information is protected so that only authorised entities can access it.
- **Integrity:** Ensure that the message received is the same as the message sent.
- **Authenticity:** Ensure that the origin of a message can be identified.
- **Non-repudiation:** Ensure that an entity who signed a message cannot successfully deny later that they did so.

Asymmetric Cryptography

- Private key
  - Has to be kept secret
  - Used for signing, decryption, etc.
- Public key
  - Can be shared / distributed
  - Often associated to an identity
  - Used for verification, encryption, etc.

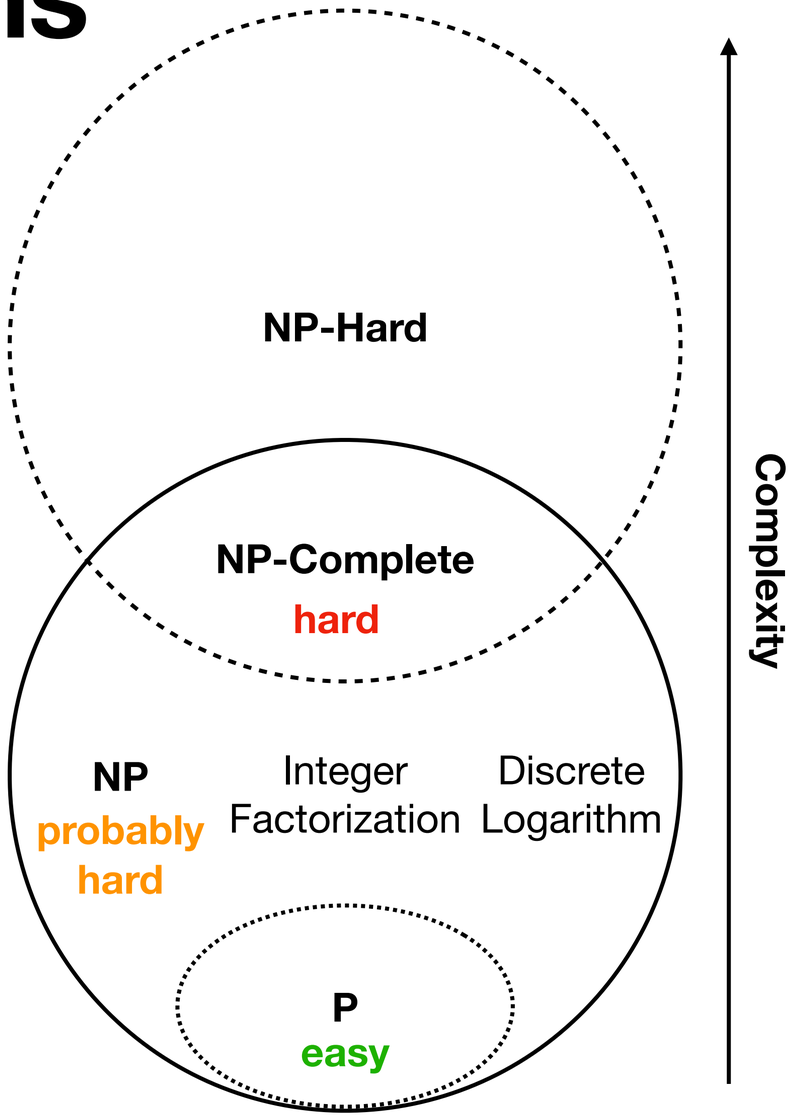


- It’s easy to compute the public from the private key but hard the other way around. But what does that actually mean?

Computationally Hard Problems

Computationally Hard Problems

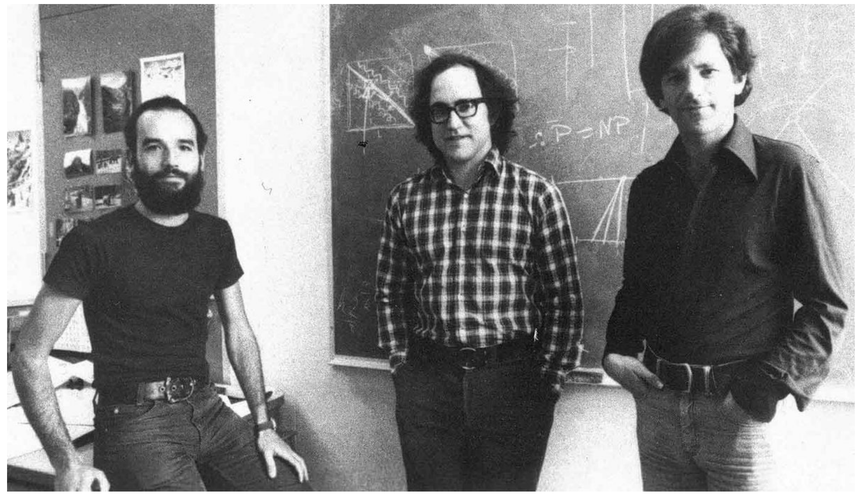
- Are there any computational problems which cannot be solved efficiently (as far as we know)?
- **Integer Factorization (IF)**
  - Given integer n = pq, find primes p and q.
  - Assumption behind RSA-based cryptography.
- **Discrete Logarithm (DL)**
  - Given group elements g and h, find a positive integer x such that $h = g^x$.
  - Assumption behind Elliptic Curve Cryptography.



The Rivest-Shamir-Adleman (RSA) Trapdoor Permutation

- Can we turn the IFP into a construction that is useful for cryptography?
- Let n = pq for primes p and q and let e be an integer such that gcd(e, φ(n)) = 1 where φ(n) = (p-1)(q-1).
- If p and q are kept secret, then it is hard (computationally infeasible) to compute φ(n) and thus it is hard to compute d = e^{-1} mod φ(n) from e.
- Without knowing (p,q,d) it is hard to invert the RSA trapdoor permutation

$$f(x) = x^e \bmod n$$

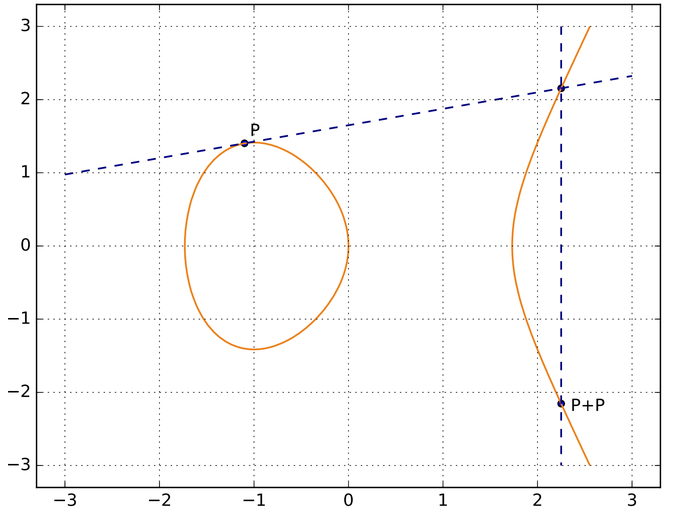


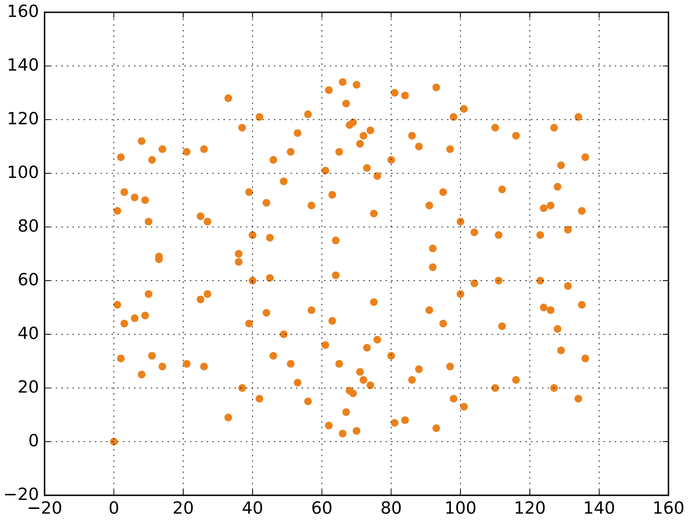
Variants of the Discrete Logarithm Problem

- Computational Diffie-Hellman (CDH) problem:
  - Given g, g^x, g^y for randomly chosen x, y, can you find g^{xy}?
- Decisional Diffie-Hellman (DDH) problem:
  - Can you distinguish (g^x, g^y, g^{xy}) from (g^x, g^y, g^z) for randomly chosen x, y, z?
- Relevant when theoretically proving the security of crypto schemes.
- There are many other similar assumptions.
- Not needed for this training but good to have seen it / be aware of.

Elliptic Curves

- Mathematical object that can be used for cryptography: DLP is considered hard
- Various forms:
  - Weierstrass: $y^2 = x^3 + ax + b$
  - Montgomery: $dy^2 = x^3 + ax^2 + x$ (e.g. Curve25519, Curve448)
  - (Twisted) Edwards: $ax^2 + y^2 = 1 + dx^2y^2$ (e.g., Edwards25519)
  - BLS (pairing-friendly): $y^2 = x^3 + b$ (e.g., BLS12-381)
- Standard for DL-based crypto nowadays
- Basis for many advanced crypto schemes (BLS signatures, certain ZKPs, etc.)

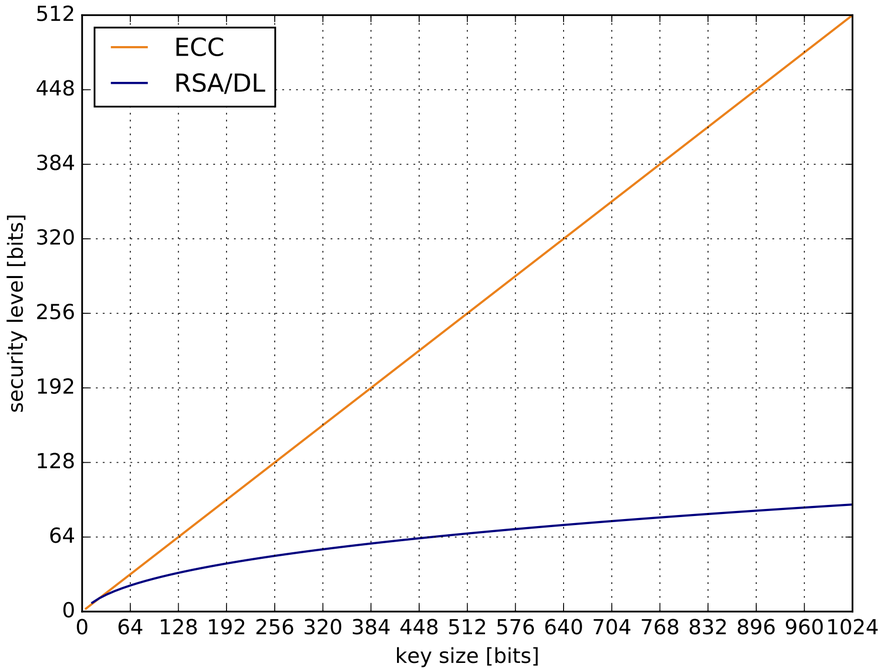




RSA/DL vs. Elliptic Curves

- Elliptic curves provide higher security per bit than RSA/DL.

	Key Size	
Security Level	**RSA/DL**	**ECC**
128	2048	256
256	15360	512



- ECC can use smaller keys and thus has usually better performance.

Public Key Encryption

Public Key Encryption Schemes

- KeyGen(λ) → (sk, pk)
  - Takes a input a *security parameter* λ and generates a key pair consisting of a *private key sk* and a *public key pk.*
- Encrypt(pk, m) → c
  - Takes as input the *recipient's public key pk* and a *message m* and returns the corresponding *ciphertext c.*
- Decrypt(sk, c) → m
  - Takes as input the *recipient's secret key sk* and the ciphertext and returns the corresponding message m.

RSA Textbook Encryption

- KeyGen(λ) → (sk, pk)
  - Pick secure primes p and q and integer e, compute n := pq and integer d := e^{-1} mod φ(n).
  - Return (sk, pk) with sk := (p, q, d) and pk := (n, e).
- Encrypt(pk, m) → c
  - Parse (n, e) = pk.
  - Return c := m^e mod n.
- Decrypt(sk, c) → m
  - Parse (p, q, d) = sk.
  - Return m = c^d mod n.

Is this secure?

Attacks on RSA Textbook Encryption

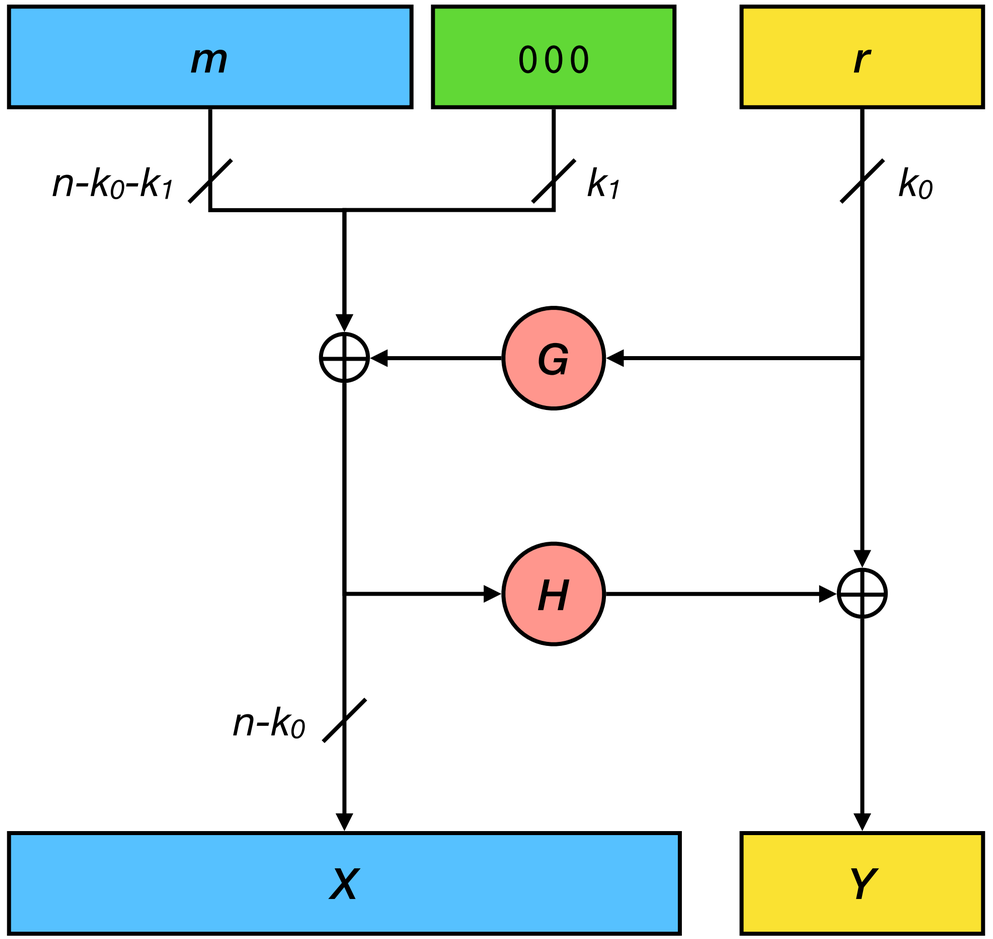
- Since encryption is deterministic, an attacker can brute-force the message knowing the ciphertext if the message space is small.
- If e and m are small such that $m^e < n$ decryption is possible without knowing d.
- If e = 3 and we encrypt the same message m under different keys (n_1, e), (n_2, e), (n_3, e), we can recover the message from the resulting ciphertexts.
- Common modulus attack: assume key pairs $sk_i = (N, d_i)$, $pk_i = (N, e_i)$.
  - Any person having one of these key pairs can read anything encrypted with any of the public keys.
  - If the same message is encrypted with any two public keys, an attacker who sees the ciphertexts and knows the public keys can recover the message.
- And so on …

RSA Optimal Asymmetric Encryption Padding

RSA-OAEP: secure RSA-based encryption scheme.

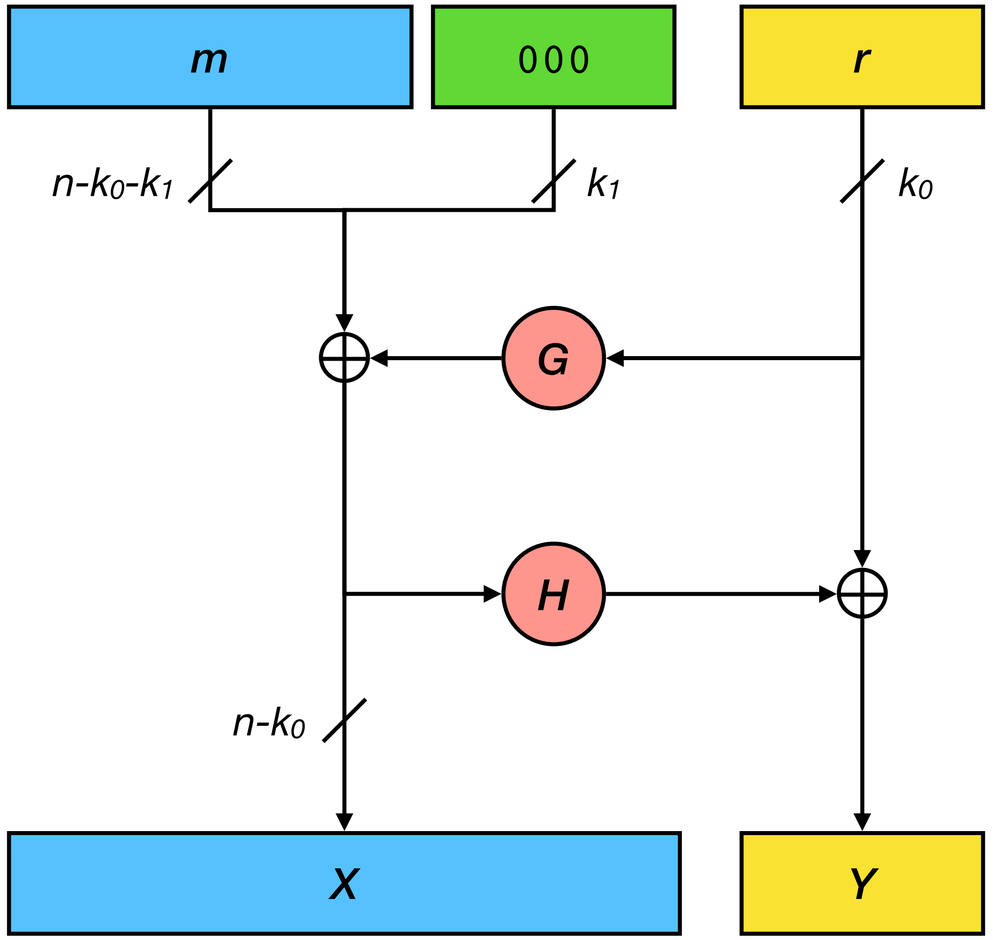
Let k_0 and k_1 be positive integers and let G and H be two cryptographic hash functions.

- KeyGen(λ) → (sk, pk)
  - Compute key pair as with textbook RSA.
- Encrypt(pk, m) → c
  - Pick a random k_0-bit bitstring r and pad m' = m || 0^{k1}.
  - Compute x = G(r) ⊕ m' and y = r ⊕ H(x) and return c = (x || y)^e mod n.
- Decrypt(sk, c) → m
  - Compute x || y = c^d mod n, r = y ⊕ H(x), and m' = G(r) ⊕ x.
  - Check if m' = m || 0^{k1} for some m and if so return m.



RSA Optimal Asymmetric Encryption Padding

- Secure against adaptive chosen ciphertext attacks (IND-CCA2)
- PKCS#1 v2.1 / v2.2 padding (no Bleichenbacher attack)
- Randomized (via r)
- Two hash function calls (G and H)
- Messages of max. 1520 bits with RSA-2048 and SHA256
- More information:
  - https://tools.ietf.org/html/rfc4055
  - https://tools.ietf.org/html/rfc5756



ElGamal Encryption

Let $\mathbb{G}$ be a cyclic group of prime order q with generator g.

- KeyGen(λ) → (sk, pk)
  - Choose a random integer x from {1,…,q-1} and return (sk, pk) := (x, g^x).
- Encrypt(pk, m) → c
  - Map m to a group element in $\mathbb{G}$.
  - Choose a random integer r from {1,…,q-1} and return (c_1,c_2) := (g^r, m × pk^r).
- Decrypt(sk, c) → m
  - Return m = c_1^{-x} × c_2.

ElGamal Encryption

- Secure against chosen plaintext attacks (IND-CPA)
- Unconditionally malleable:
  - Let (c_1, c_2) be a ciphertext for an unknown message m.
  - Then $(c_1, k \times c_2)$ is a valid ciphertext for $k \times m$.
- Often used in a hybrid manner (e.g., in PGP):
  - Symmetric encryption for message.
  - ElGamal encryption for symmetric key.

Other Public Key Encryption Schemes

- Cramer-Shoup:
  - Non-malleable extension of ElGamal.
  - Secure against adaptive chosen ciphertext attacks (IND-CCA2)
- Pallier:
  - Additively homomorphic cryptosystem:

$$\text{Enc}(pk, m_1) \times \text{Enc}(pk, m_2) = \text{Enc}(pk, m_1 + m_2).$$

  - Relies on the *decisional composite residuosity assumption.*
  - Secure against chosen plaintext attacks (IND-CPA), can be extended to IND-CCA2.
  - Sometimes used in e-voting and e-cash schemes.

Digital Signatures

Digital Signature Schemes

- KeyGen(λ) → (sk, pk)
  - Takes as input a *security parameter* λ and generates a key pair consisting of a *private key sk* and a *public key pk.*
- Sign(sk, m) → σ
  - Takes as input the sender's secret key sk and a message m and produces a *signature* σ.
- Verify(pk, m, σ) → 0/1
  - Takes as input the sender's public key pk, a signature σ, and a message m and returns 1 if the signature σ on message m is correct and 0 otherwise.

RSA Textbook Signature Scheme

- KeyGen(λ) → (sk, pk)
  - Pick secure primes p and q and integer e, compute n := pq and integer d := e^{-1} mod φ(n).
  - Return (sk, pk) with sk := (p, q, d) and pk := (n, e).
- Sign(sk, m) → σ
  - Parse sk = (p, q, d).
  - Return $\sigma := m^d \bmod n$
- Verify(pk, m, σ) → 0/1
  - Parse pk = (n, e).
  - Return 1 if $m = \sigma^e \bmod n$ and 0 otherwise.

Is this secure?

Attacks on RSA Textbook Signature Scheme

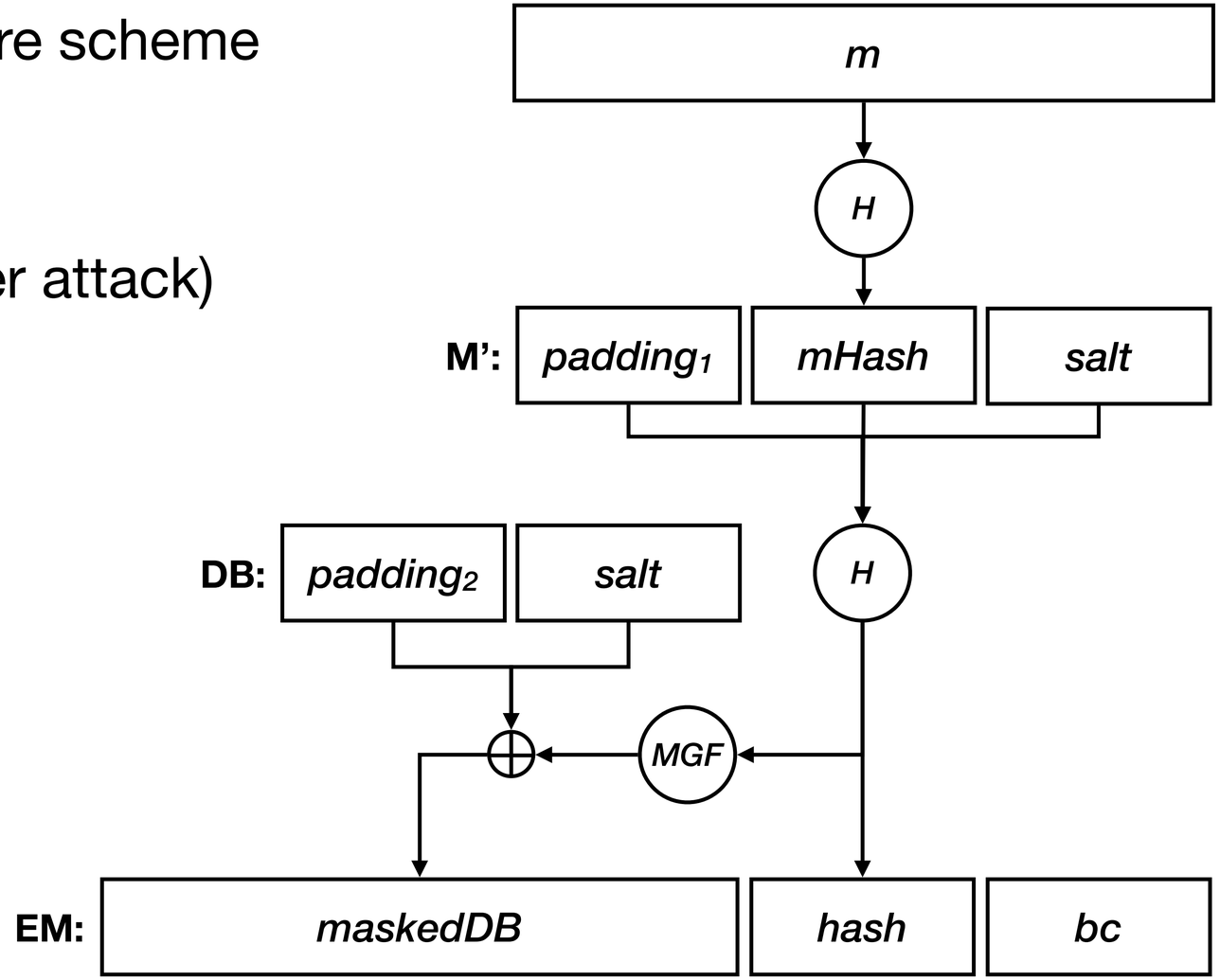
- There are various attacks on the RSA textbook signature scheme which all exploit its homomorphic properties in one way or another.
- Blinding Attack:
  - Given message m, choose a value r.
  - Attacker computes blinded message $m' = r^e m$.
  - Signer computes signature on m': $\sigma' = (m')^d \bmod n$
  - Now attacker *also* gets a signature σ on m:

$\sigma = r^{-1}\sigma' \bmod n = r^{-1}(m')^d \bmod n = r^{-1}(r^e m)^d \bmod n = r^{-1} r m^d \bmod n = m^d \bmod n$

- And so on …

RSA Probabilistic Signature Scheme

- RSA-PSS: secure RSA-based signature scheme
- Proven semantically secure
- PKCS#1v2.1 / v2.2 (no Bleichenbacher attack)
- Randomized encryption
- Three hash function calls
- More information:
  - https://tools.ietf.org/html/rfc4055
  - https://tools.ietf.org/html/rfc5756



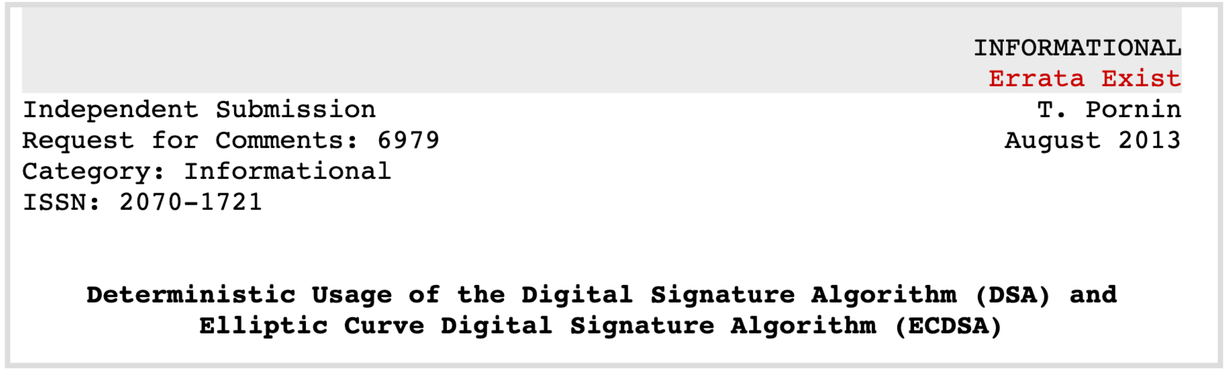
(EC)DSA Signature Scheme

Let $\mathbb{G}$ be a cyclic group of prime order q with generator g and let H be a cryptographic hash function.

- KeyGen(λ) → (sk, pk)
  - Choose a random integer x from {1,…,q-1} and return (sk, pk) := (x, g^x).
- Sign(sk, m) → σ
  - Choose a random integer k from {1,…,q-1}.
  - Compute r := g^k and s := k^{-1} × (H(m) + r × sk) and return σ := (r, s).
- Verify(pk, m, σ) → 0/1
  - Parse := (r, s) = σ.
  - Compute w := s^{-1}, u_1 := H(m) × w, and u_2 := r × w.
  - Return 1 if r = (g^{u1} × pk^{u2}) and 0 otherwise.

Fragility of (EC)DSA

- Random k must be unique and secret, otherwise the private key can be recovered.
- Defense: set k to a hash of the message and private key,
  (see https://tools.ietf.org/html/rfc6979).



INFORMATIONAL
Errata Exist

Independent Submission — T. Pornin
Request for Comments: 6979 — August 2013
Category: Informational
ISSN: 2070-1721

Deterministic Usage of the Digital Signature Algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA)

- Similar approach in **EdDSA** (Edwards-curve DSA):
  - ECC signatures using the Schnorr signature technique, simpler than ECDSA.
  - **Ed25519** now widely used (OpenSSH, OpenSSL, GnuPG, etc.).

Crypto Fail Tales: The PS3 Hack

Hackers Describe PS3 Security As Epic Fail, Gain Unrestricted Access

BY MIKE BENDEL DECEMBER 29, 2010 @ 11:19 AM

Prominent hackers Bushing, Marcan, and Sven took the stage at this year's annual Chaos Communication Congress (27C3) to showcase their latest underground efforts on PS3. The trio describe Sony's security measures as an 'epic fail,' pointing to the botched implementation of ECDSA. Apparently, the so-called 'random' number used to create the private key is always static.

Crypto Fail Tales: The PS3 Hack

- Assume randomness k is used to sign two different messages m_1 and m_2:
  - $\sigma_1 = k^{-1} \times (H(m_1) + r \times \mathbf{sk})$
  - $\sigma_2 = k^{-1} \times (H(m_2) + r \times \mathbf{sk})$
- Then:
  - $\sigma_1 - \sigma_2 = k^{-1} \times (H(m_1) + r \times \mathbf{sk}) - k^{-1} \times (H(m_2) + r \times \mathbf{sk})$
  - $\sigma_1 - \sigma_2 = k^{-1} \times (H(m_1) + r \times \mathbf{sk} - H(m_2) - r \times \mathbf{sk})$
  - $\sigma_1 - \sigma_2 = k^{-1} \times (H(m_1) - H(m_2))$
  - $k = (H(m_1) - H(m_2)) \times (\sigma_1 - \sigma_2)^{-1}$
- Once you know k, you can solve one of the signature equations for secret key **sk**:
  - $\mathbf{sk} = ((\sigma \times k) - H(m)) \times r^{-1}$

Schnorr Signature Scheme

Let $\mathbb{G}$ be a cyclic group of prime order q with generator g and let H be a cryptographic hash function.

- KeyGen(λ) → (sk, pk)
  - Choose a random integer x from {1,…,q-1} and return (sk, pk) := (x, g^x).
- Sign(sk, m) → σ
  - Choose a random integer k from {1,…,q-1}.
  - Compute r = g^k, c = H(r || m) and s = k - sk × c mod q and return σ = (s, c).
- Verify(pk, m, σ) → 0/1
  - Parse (s, c) = σ.
  - Return 1 if c = H(g^s × pk^c || m) and 0 otherwise.

Note: Like in (EC)DSA, randomness k has to be fresh for every new message signed with the same key.

BLS Signature Scheme

Let $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ be cyclic groups with generators g_1, g_2, g_T, bilinear pairing operator $e: \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ and let H be a cryptographic hash function.

- KeyGen(λ) → (sk, pk)
  - Choose a random integer x from {1,…,q-1} and return (sk, pk) := (x, g_2^x).
- Sign(sk, m) → σ
  - Return σ := $H(m)^{sk}$.
- Verify(pk, m, σ) → 0/1
  - Return 1 if e(H(m), pk) = e(σ, g_2) and 0 otherwise.

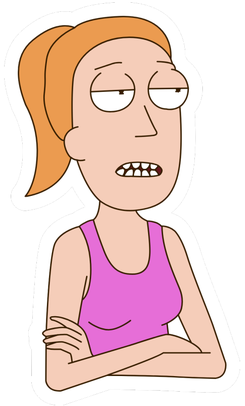
Key Exchange

Diffie-Hellman Key Exchange

Goal: Share a secret key between two parties over an insecure channel.

Let $\mathbb{G}$ be a cyclic group of prime order q with generator g.

Choose a in {1,…,q-1}



g^a

g^b

Choose b in {1,…,q-1}

Compute $(g^b)^a = g^{ab}$

Compute $(g^a)^b = g^{ab}$

Any security problems?

Man-in-the-Middle Attacks on DH

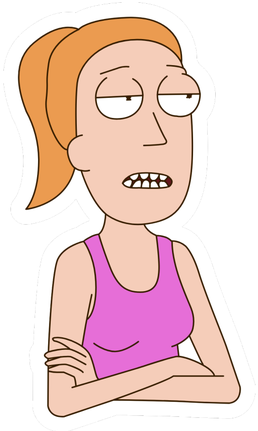
Let $\mathbb{G}$ be a cyclic group of prime order q with generator g.

Choose a in {1,…,q-1}

Choose c in {1,…,q-1}

Choose d in {1,…,q-1}

Choose b in {1,…,q-1}



g^a →

← g^c

g^b →

← g^d

Compute $(g^c)^a = g^{ac}$

Compute g^{ac} and g^{bd}

Compute $(g^d)^b = g^{bd}$

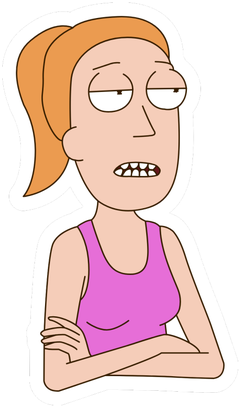
Authenticated Diffie-Hellman Key Exchange

Goal: Share a secret key between two parties over an insecure channel and authenticate them to each other

Let $\mathbb{G}$ be a cyclic group of prime order q with generator g.

Holds sk_A, pk_A

Choose a in {1,…,q-1}



$m_A := g^a$, $\sigma_A := sign(sk_A, m_A)$

$m_B := g^b$, $\sigma_B := sign(sk_B, m_B)$

verify(pk_B, m_B, σ_B) = 1

Compute $(g^b)^a = \mathbf{g^{ab}}$

Holds sk_B, pk_B

Choose b in {1,…,q-1}

verify(pk_A, m_A, σ_A) = 1

Compute $(g^a)^b = \mathbf{g^{ab}}$

ADH assumes parties know each others' long-term public keys pk_A and pk_B. So what did we win?

Hybrid Encryption

Hybrid Encryption via Key Encapsulation

Combines symmetric (SE) and asymmetric encryption (AE).

- HE.KeyGen(λ) → (sk, pk)
  - Return (sk, pk) := AE.KeyGen(λ).
- HE.Encrypt(pk, m) → c
  - Derive a fresh symmetric key k.
  - Compute c_m := SE.Encrypt(k, m) and c_k := AE.Encrypt(pk, k) and return c := (c_m, c_k).
- HE.Decrypt(sk, c) → m
  - Parse (c_m, c_k) = c.
  - Compute k = AE.Decrypt(sk, c_k) and return m = SE.Decrypt(k, c_m).

Hybrid Encryption: Integrated Encryption Scheme (IES)

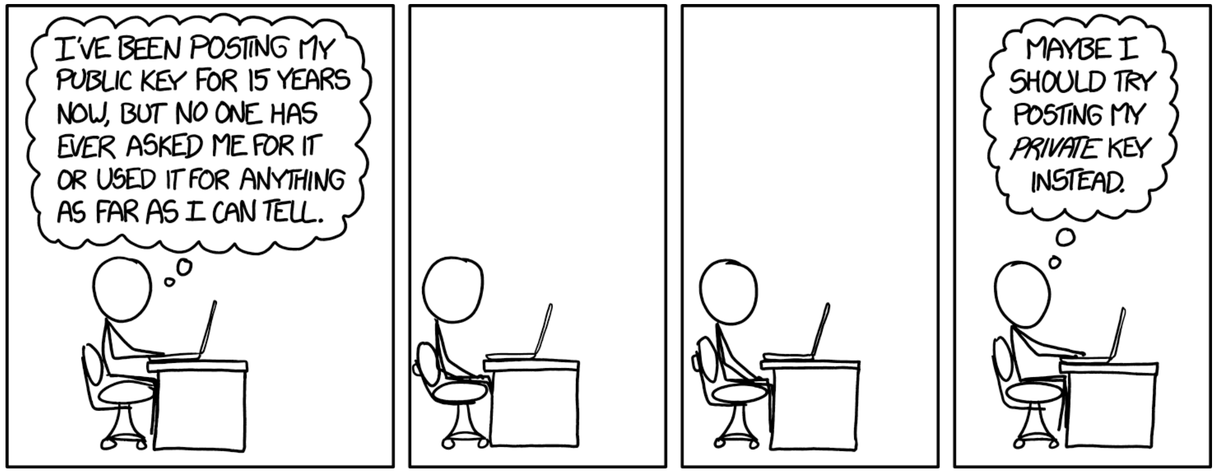
Combines symmetric encryption (SE) and Diffie-Hellman (DH) key exchange.

- HE.KeyGen(λ) → (sk, pk)
  - Return (sk, pk) := (x, g^x).
- HE.Encrypt(pk, m) → c
  - Parse g^x = pk and derive a fresh DH key share y.
  - Compute $pk^y = g^{xy}$, k = KDF(g^{xy}), c_m := SE.Encrypt(k, m), and return c := (g^y, c_m).
- HE.Decrypt(sk, c) → m
  - Parse (g^y, c_m) = c.
  - Compute $(g^y)^{sk} = g^{xy}$, k = KDF(g^{xy}), m = SE.Decrypt(sk, c_m), and return m.

Summary

Summary

- Public key cryptography: crucial tool to secure modern communication
- Encryption: RSA-OAEP, ElGamal, Cramer-Shoup, Pallier
- Signing: RSA-PSS, ECDSA, EdDSA, Schnorr, BLS
- Key exchange: DH, ADH
- Hybrid encryption:
  - Symmetric encryption + KEM
  - Symmetric encryption + DH (IES)



https://xkcd.com/1553/

crypto attacks & defenses

RELOADED

JP Aumasson, Philipp Jovanovic

ringzerø

public-key crypto