CS 6160 Cryptology Lecture 16: Public-Key Encryption Schemes & Digital Signatures

Maria Francis

November 10, 2020

Hybrid Encryption

- We have seen that a CPA-secure PKE for ℓ'-bit messages can be used to obtain a CPA-secure PKE for messages of arbitrary length.
- If we want to encrypt ℓ-bit messages that will mean $\gamma = \lceil \ell/\ell' \rceil$ calls to the original PKE and that means more computation time and ciphertext length.
- What if we do SKE in tandem with PKE since SKE is faster and has lower ciphertext expansion.
- We have hybrid encryption: Use PKE to obtain a shared key k and encrypt m using a SKE and key k.
- Note: The SKE is used as a component but it is a full-fledged *public-key encryption scheme* because the sender and receiver do not share any secret key in advance.
- We skip the security details and the speedup calculations.

El Gamal Encryption

- We have seen abstractly how PKE works but not really seen the construction of one.
- We first look at El Gamal encryption based on the Diffie-Hellman protocol.

Lemma

Let G be a finite group and let $m \in G$ be arbitrary. Then choosing uniform $k \in G$ and setting $k' := k \cdot m$ gives the same distribution for k' as choosing uniform $k' \in G$. I.e., for any $\hat{g} \in G$, we have

$$Pr[k \cdot m = \hat{g}] = 1/|G|.$$

Proof : $Pr[k \cdot m = \hat{g}] = Pr[k = \hat{g} \cdot m^{-1}]$. k is uniform and so prob. of k equal to some element is exactly $1/|G|$.

Outline

- If we have a shared key k between Alice and Bob then to encrypt $m \in G$ Alice just has to compute $k^{'} = k \cdot m$.
- Bob gets m by doing $k^{'}/k$ – the lemma ensures that perfect secrecy happens.
- This is like OTP where the group operation was bit-wise XOR and group is the set of strings of fixed length.
- How to get a shared random-looking k over a public channel - Diffie-Hellman protocol.
- In El Gamal, $\mathcal{G}$ outputs a cyclic group G with order q, $||q|| = n$ and a generator g in poly-time.

El Gamal Algorithm

1. $Gen(1^n)$ runs $\mathcal{G}$ to get (G, q, g). Then choose a uniform $x \in \mathbb{Z}_q$ and compute $h := g^x$. $pk = (G, q, g, h)$ and $sk = (G, q, g, x)$. Message space is G.
2. $Enc(pk, m)$: Chooses uniform $y \in \mathbb{Z}_q$ and outputs $\langle c_1, c_2 \rangle = \langle g^y, h^y \cdot m \rangle$.
3. $Dec(sk, \langle c_1, c_2 \rangle)$: $m := c_2 / {c_1}^x$.

Correctness:

$$m = \frac{c_2}{c_1^x} = \frac{h^y m}{(g^y)^x}$$

$$\frac{(g^x)^y m}{g^{xy}} = m.$$

Security Proof

If the DDH problem is hard relative to $\mathcal{G}$ then the El Gamal encryption scheme is CPA-secure.

Proof is on similar lines to the Problem 4 that we discussed in the practice question session last week. Replace the secure key-exchange protocol with Diffie-Hellman protocol and you are done. The details of the proof is also there in the textbook.

El Gamal Implementation Issues

- Sharing public parameters about the group: usually fixed once using NIST standards and then shared by multiple receivers.
- Choice of group: q is prime and elliptic curve groups are popular.
- Message space: It is a group G and not any bit string. Not very convenient.
  - In some cases, messages can be encoded as group elements.
  - Often what is done is to use a hybrid encryption scheme.
  - A group element m is chosen and encrypted using El Gamal and the actual message is encrypted using a SKE that uses the key $H(m)$ where H is an appropriate key-derivation function.

Malleability of El Gamal

- Consider $\mathcal{A}$ who intercepts a $c = \langle c_1, c_2 \rangle$ encrypted using $pk = \langle G, q, g, h \rangle$.
- $\mathcal{A}$ constructs $c' = \langle c_1, c_2' \rangle$ where $c_2' = c_2 \alpha$ for some $\alpha \in G$.
- If c corresponded to m (unknown to $\mathcal{A}$) we have

$$c_1 = g^y \text{ and } c_2 = h^y m.$$

- But then,

$$c_1 = g^y \text{ and } c_2' = h^y(\alpha m).$$

  Thus c' is a valid encryption of $\alpha \cdot m$.
- $\mathcal{A}$ can transform an encryption of the (unknown) message m into an encryption of the (unknown) message $\alpha \cdot m$.

CCA-security & El Gamal

- The hybrid encryption scheme we discussed earlier where the key-derivation function H is modeled as random oracle makes El Gamal CCA-secure.
- The security relies on the so-called gap-CDH assumption.
- It says informally the CDH problem remains hard even given an oracle that solves the DDH problem.
- This is true for the groups we discussed so far.
- Typically, assuming DDH problem alone gives us CPA-security and the stronger gap-CDH assumption is needed to obtain CCA-security.
- Interesting read: DHIES (Diffie-Hellman Integrated Encryption Scheme)/ECIES (Elliptic Curve Integrated Encryption Scheme) CCA-secure variants included in the ISO/IEC 18033-2 standard.

Plain RSA

- Based on RSA asumption we saw earlier.
- GenRSA: a PPT algo that outputs a modulus N a product of two n-bit primes (using GenModulus) along with e, d satisfying $ed = 1 \bmod \varphi(N)$.
- RSA encryption relies on the fact that knowing d one can recover m from c efficiently: $c^d \bmod N$.
- Also, without knowledge of d even knowing N and e no one can recover m from c efficiently, *at least if m is chosen uniformly from $\mathbb{Z}_N^*$.*
- Plain RSA is just that algorithm :
  - Gen : using GenRSA obtains N, e, d. Set $pk = \langle N, e \rangle$ and $sk = \langle N, d \rangle$.
  - Enc: $c = m^e \bmod N$
  - Dec: $m = c^d \bmod N$

Attacks on Plain RSA: Recovery of m

- It is deterministic encryption. This makes many attacks possible.
- An attack that gives a quadratic improvement in recovering m.

  - Let $m < 2^n$ and $\alpha \in (\frac{1}{2}, 1)$ be some constant.
  - We have $pk = N, e$ and c.
  - Set $T = 2^{\alpha n}$.
  - For $r = 1$ to T : set $x_r := \frac{c}{r^e} \bmod N$.
  - Sort the pairs $\{(r, x_r)\}_{r=1}^{T}$ based on x_r.
  - For $s = 1$ to T: check if $x_r = s^e \bmod N$ for some r.
  - If yes, then break and return $r \cdot s \bmod N$.

Analysis of recovery of m attack

- Time complexity:
  - Time to sort $2^{\alpha n}$ pairs (r, x_r) takes $\mathcal{O}(n2^{\alpha n})$.
  - Binary search for the final search for r s.t. $x_r = s^e \bmod N$.
- Correctness:
- Assumption: For appropriate $\alpha > 1/2$ if m is a uniform n-bit integer then with high prob. there exist r, s with $1 < r \leq s \leq 2^{\alpha n}$ s.t. $m = rs$.
- Since $c = m^e = (r \cdot s)^e = r^e \cdot s^e \bmod N$ and $x_r = c/r^e = s^e \bmod N$, with $r, s < T$, algorithm finds m.

Attack on short messages

- If $m < N^{1/e}$ raising m to the eth power modulo N involves no modular reduction, i.e $m^e \mod N = m^e$.
- Given c of m then $\mathcal{A}$ can determine m by computing $m := c^{1/e}$ over integers not modulo N.
- This can be done efficiently in $\mathrm{poly}(||N||)$ time since finding eth roots over integers is easy.
- For small e this is a serious weakness. Let $e = 3$ and assume $||N|| \approx 1024$ bits then the attack works even when m is a 300 bit integer.
- We look one more attack – broadcast attack, when we send same message to multiple receivers.
- There are more attacks possible and the textbook is a good reading for a few more attacks.

Chinese Remainder Theorem

Theorem

Let $m_1, m_2, \ldots, m_n$ be integers s.t. $gcd(m_i, m_j) = 1$, $i \neq j$. Let $a_1, a_2, \ldots, a_k \in \mathbb{Z}$. Then the system of simultaneous congruences,

$$x = a_1 \mod m_1$$
$$x = a_2 \mod m_2$$
$$\cdots,$$
$$x = a_k \mod m_k$$

has a solution $x = c$.

Further, if $x = c$ and $x = c'$ are both solutions, then $c \equiv c' \mod m_1 m_2 \cdots m_k$.

Chinese Remainder Theorem

- The first recorded instance of the problem of this type appears in a Chinese mathematical work from late 3rd or early 4th century.
- Sun Tzu Suan Ching book: *We have a number of things but we do not know exactly how many. If we count them by threes we have two left over, count them by fives, we have three left over. If we count them by sevens, we have two left over. How many things are there?*
- The proof was not given clearly there.

Chinese Remainder Theorem - Idea of the Proof

We look for an integer x that simultaneously solves both of the congruences

$$x = 1 \mod 5$$
$$x = 9 \mod 11.$$

- The first equation gives us $x = 1 + 5y$, $y \in \mathbb{Z}$. Substituting into second, $1 + 5y = 9 \mod 11$ and hence $5y = 8 \mod 11$.
- Multiplying with 5^{-1} $(= 9)$ on both sides we get $9 \cdot 8 = 72 = 6 \mod 11$. Substituting for y we get $x = 1 + 5 \cdot 6 = 31$.

Hastad's broadcast attack on RSA

- Bob has 3 different public keys $pk_1 = (N_1, e)$, $pk_2 = (N_2, e)$, $pk_3 = (N_3, e)$, with $e = 3$ and $gcd(N_i, N_j) = 1$, $1 \leq i < j \leq 3$, corresponding to three recipients.
- Bob wants to send m, $m < \min(N_1, N_2, N_3)$. He encrypts m to get c_i corresponding to pk_i, $1 \leq i \leq 3$.
- Eve is eavesdropping and knows $c_1 = m^3 \bmod N_1$, $c_2 = m^3 \bmod N_2$, $c_3 = m^3 \bmod N_3$.
- Since $gcd(N_i, N_j) = 1$, Eve can apply CRT to get a $c^{'}$ such that $c^{'} \equiv M^3 \bmod N_1 N_2 N_3$.
- Since $M \lneq N_i$ for $1 \leq i < j \leq 3$, this implies $M^3 \lneq N_1 N_2 N_3$ and therefore $c = \sqrt[3]{c^{'}} = m$ is the plaintext message.

Padded RSA

- How to make the mapping randomized so that encryption is not deterministic?
- Randomly pad the message before encrypting.
- Sender chooses a uniform bit-string $r \in \{0,1\}^{\ell}$ and set $\hat{m} = r \circ m$.
  - Gen is the same as plain RSA.
  - Enc: $m \in \{0,1\}^{||N||-\ell(n)-2}$, choose a uniform $r \in \{0,1\}^{\ell}(n)$ and $\hat{m} = r \circ m$.
  - Output $c := \hat{m}^e \bmod N$.
  - Dec: $\hat{m} = c^d \bmod N$ and output the $||N|| - \ell(n) - 2$ LSB of $\hat{m}$.
- There are standards based on the padded RSA which are CPA-secure but if the padding is too small then it is not CPA-secure.

Malleability of RSA

- The guarantee we have is : if m is uniform then Eve on having c and N, e cannot recover m
- That is a weak guarantee. Not even CPA-secure.
- RSA is malleable making it not CCA-secure as well.
  - Given $c = m^e \bmod N$ of unknown m we can generate c', encryption of $2m$.

$$\begin{aligned} c' &= 2^e c \bmod N \\ &= 2^e \cdot m^e = (2m)^e \bmod N. \end{aligned}$$

RSA-OAEP

- RSA-based CCA-secure encryption: optimal asymmetric encryption padding (OAEP).
- It is padding that we saw before but more complex.
- $\ell(n), k_0(n), k_1(n)$: integer-valued functions s.t. $k_0(n), k_1(n) = \Theta(n)$ and $\ell(n) + k_0(n) + k_1(n) <$ min. bit-length of modulus output of GenRSA.
- We fix n and refer to these functions without the parameter n.
- Consider two hash functions modeled as independent random oracles:

$$G : \{0,1\}^{k_0} \to \{0,1\}^{\ell+k_1}, \quad H : \{0,1\}^{\ell+k_1} \to \{0,1\}^{k_0}$$

RSA-OAEP

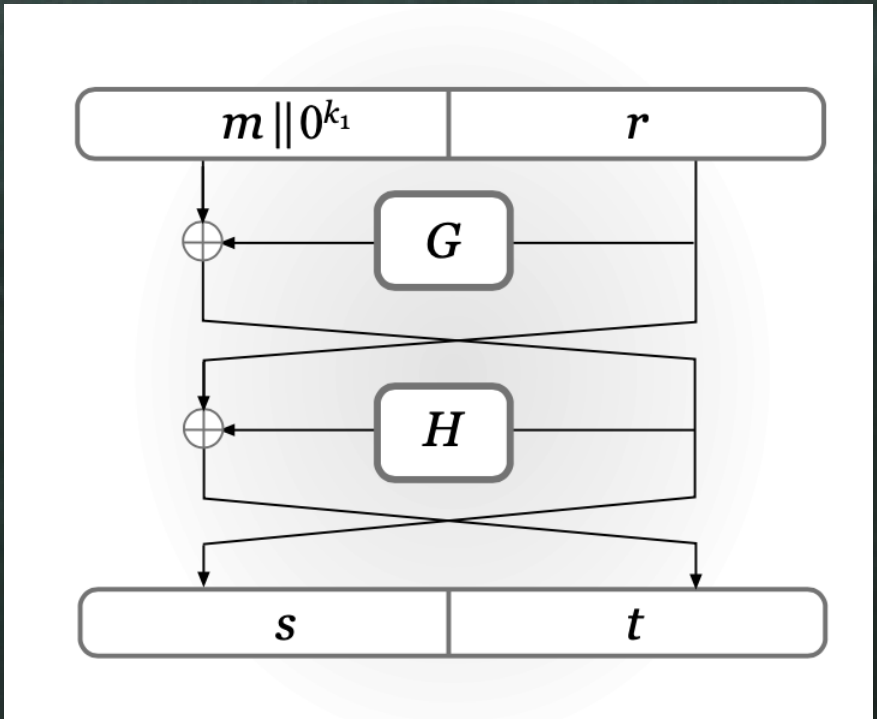
- The padding is done through a 2-round Feistel network with G and H as round functions.
  - Set $m' := m \circ 0^{k_1}$, $m \in \{0,1\}^{\ell}$ and choose $r \in \{0,1\}^{k_0}$.
  - Compute:

$$s := m' \oplus G(r) \in \{0,1\}^{\ell+k_1}, \quad t := r \oplus H(s) \in \{0,1\}^{k_0}$$

    and set $\hat{m} := s \circ t$.

- Encryption is done on $\hat{m}$ and to decrypt it obtains $c^d \bmod N$ and inverts the Feistel network.
- Once obtaining m' one has to verify if the k_1 LSBs are 0s, else reject c.

OAEP



The formal construction of the OAEP mechanism is given in the textbook (Construction 11.36).

Intuition for CCA-Security

- For a uniform r, $s := m' \oplus G(r)$, $t = r \oplus H(s)$, $c = (s \circ t)^e \bmod N$.
- If $\mathcal{A}$ never queries r to G, $G(r)$ is uniform and m is uniform.
- Can $\mathcal{A}$ query r to G?
  - Value of r is XORed with $H(s)$ so$\mathcal{A}$ needs to query s to H.
  - Instead $\mathcal{A}$ just guessed r : that happens with negl. prob. if $||r|| = k_0$ is sufficiently long.
- What about querying s to H?
  - Compute s from $s \circ t \bmod N$.
  - Not the RSA problem since that would mean computing s and t. But still not easy for the right parameters.
- Formal proof has many aspects.
  - If c was not constructed legally then the chances of k_1 LSBs being 0 is negligible.
  - Else S.T. $\mathcal{A}$ learns nothing from querying the decryption oracle.

Digital Signatures

- MAC analogue in public-key setting (with many key differences), for integrity/authenticity.
- Here, owner of the *pk* is the sender, signs the message with private key.
- What are the differences w.r.t. MACs?
  - Digital signatures are publicly verifiable. I.e. the same signed message can verified by multiple Bobs.
  - This implies transferability: I.e. third parties can verify themselves.
  - Important for certificates and public-key infrastructures.
  - Non-repudiation: The signed S cannot later deny (to say a judge) he authenticated the message (say a contract).
- What about considering signatures as an inverse of encryption schemes?
  - Not true, some cases it cannot be done and other cases it can lead to insecure signature schemes.

Formal Definition

- Just like MACs: Three PPT algorithms ($Gen, Sign, Verify$).
- $Gen(1^n)$ outputs (pk, sk).
- $Sign(sk, m)$ and outputs signature σ. $\sigma \leftarrow Sign_{sk}(m)$.
- Deterministic $Verify$ takes pk, m and σ as input and outputs a bit b where $b = 1$ implies valid signature else invalid signature. $b := Verify_{pk}(m, \sigma)$.

Formal Definition Contd.

- We need *Verify* to output 1 for all legal messages except with negligible probability over (pk, sk).
- If there is a function ℓ s.t. for every (pk, sk) output by $Gen(1^n)$ the message space is $\{0,1\}^{\ell(n)}$, then we say that $(Gen, Sign, Verify)$ is a signature scheme for messages of length $\ell(n)$.
- We have already discussed that reliable distribution of pk is a problem but signature schemes means it need to be carried out only once.
- Security of signature scheme is defined similar to the MAC scheme: $\mathcal{A}$ should be unable to output a forgery even if it obtains signatures on many other messages of its choice.
- We call the analogous experiment $Sig - forge$ and signature schemes are secure if $Pr[Sig - forge_{\mathcal{A},\Pi}(1^n) = 1] \leq \mathrm{negl}(n)$.

Hash-and-Sign Paradigm

- Just like PKE, signature schemes are orders of magnitude slower than MACs.
- We try to address that with hash-and-sign schemes.
- If we have a signature scheme for messages of length ℓ and we want to sign an arbitrarily long message $m \in \{0,1\}^*$.
- Use hash function H to hash m to a fixed-length digest of length ℓ and then sign digest.
- $\sigma \leftarrow Sign'_{sk}(H^s(m))$ and $Verify'_{pk}(H^s(m), \sigma) = 1$?
- If Π is a secure signature scheme for messages of length ℓ and Π_H is collision-resistant then hash-and-sign scheme is a secure signature scheme for arbitrary-length messages.

RSA signatures - Plain RSA

- *Sign*: $\sigma = m^d \bmod N$ and *Verify* outputs 1 iff $m = \sigma^e \bmod N$.
- This looks secure but it is not!
- Problem is the hardness assumption is for computing a signature for a uniform message and not for a nonuniform one or a choice of the adversary.
- Also RSA assumption does not take care of the possibility that $\mathcal{A}$ may have the signatures on other messages.
- One simple forgery: Choose a uniform $\sigma \in \mathbb{Z}_N^*$ and compute $m = \sigma^e \bmod N$ using pk.
- Output (m, σ) : it is a forgery! Maybe not a realistic attack but still a forgery!

Another attack

- $\mathcal{A}$ obtains two signatures and with that *he can output a forged signature on any message of its choice.*
- $\mathcal{A}$ wants signature on $m \in \mathbb{Z}_N^*$. He chooses $m_1, m_2 \in \mathbb{Z}_N^*$ s.t. $m = m_1 \cdot m_2 \bmod N$.
- It obtains signatures σ_1, σ_2 on m_1, m_2. $\mathcal{A}$ outputs $\sigma = \sigma_1 \cdot \sigma_2 \bmod N$ as a valid signature on m.

$$\sigma^e = (\sigma_1 \sigma_2)^e = (m_1^d \cdot m_2^d)^e = m_1 \cdot m_2 = m \bmod N.$$

RSA-FDH

- Transform the message before signing them. Done using a deterministic function H that has some properties.
- RSA-FDH signature scheme:
  1. $Gen(1^n)$: outputs N, e, d.
  2. $Sign_{sk}(m)$ gives the signature as $H(m)^d \mod N$.
  3. $Verify_{pk}(m, \sigma)$ outputs 1 iff $\sigma^e = H(m) \mod N$.
- What should ideally be H?
  1. Hard to invert.
  2. Should not admit multiplicative relations.
  3. Collision resistant.

RSA-FDH

- If you assume H is a random oracle that maps uniformly onto $\mathbb{Z}_N^*$ all the required properties are satisfied and then we call this case as RSA full-domain hash or RSA-FDH signature scheme.
- If the RSA problem is hard relative to GenRSA and H is modeled as a random oracle, then the construction we saw is secure.
- Proof we will discuss in the tutorial session.
- Signatures from discrete-log problem is not as easy to see as from the RSA assumption. We need to define identification schemes (Schnorr in particular) and use the Fiat-Shamir transform to convert them to signature schemes in the RO-model. We do not cover this in our course.