CS 120/CSCI E-177: Introduction to Cryptography

Salil Vadhan and Alon Rosen Nov. 16, 2006

Lecture Notes 15:

Public-Key Encryption in Practice

Recommended Reading.

- Katz–Lindell, Sections 9.4, 9.5.3

1 Public-Key Encryption in Practice

- All known public-key encryption schemes much slower than private-key ones and have much larger keys.
  - e.g. Plain RSA: 1000× slower than DES in hardware, and 100× slower in software (for 512-bit modulus).
  - Mainly used to exchange a "session key" for a private-key encryption scheme $\rightsquigarrow$ *hybrid encryption* (see Katz–Lindell §9.4)
- RSA overwhelmingly most popular (despite not having security equivalent to factoring like Rabin):
  - *Plain RSA*: insecure
  - *Padding a 1-bit message with random bits*: *provably* secure
  - *Padding longer mesages with random bits (PKCS #1 v 1.5 RSA)*: unproven but conjectured to be secure. Use a 88-bit random pad for 512-bit messages.
  - *More sophisticated padding using cryptographic hash functions (PKCS #1 v 2, OAEP)*: G is a PRG and H is a hash function. The encryption of m using a random pad R is $(m \oplus G(R) || R \oplus H(m \oplus G(R)))^e \bmod N$. This can be proven to be secure (even against chosen-ciphertext attack) in *Random Oracle Model*, which models the hash function as a "random function", which it is *not*, so this is only a heuristic argument.
  - *Encryption exponent 3*: (change RSA assumption accordingly)
- Discrete Log Based Schemes used in practice
  - Diffie–Hellman Key Exchange
    - Standardized in ANSI X9.42
    - Widely used in protocols to establish temporary keys for network communication, including SSH, HTTPS (SSL), and others.
  - ElGamal
    - No patent restrictions (RSA was patented)
    - Used in free products, e.g. GNU Privacy Guard, PGP

    - Used in threshold crypto applications requiring "distributed key generation" (we may cover this later)
  - Cramer–Shoup Encryption
    - Standardized in ISO 18033-2
    - Similar in spirit to ElGamal
    - Requires a collision-resistant cryptographic *hash function* to be discussed later in the course
    - Uses extra math (ciphertext and keys are longer) to achieve security under CCA (1998: first practical algorithm to do so based on a standard complexity assumption; the earlier Dolev-Dwork-Naor work using trapdoor permutations requires random oracles.)

2 Malleability

Informally, an encryption scheme is *malleable* if given a ciphertext c that is an encryption of a plaintext m (and the public key), one can efficiently generate a ciphertext c' that is an encryption of a transformation of m, that is, $c' \in E(f(m))$. (We recall that in a probabilistic encryption scheme a message may have many valid encryptions.) In particular this can be done without any knowledge of m or the secret key. Thus, it does not contradict the encryption scheme being secure in the sense of having indistinguishable encryptions.

Nevertheless, in some applications, malleability is a weakness. For example, in the context of a sealed-bid auction, an adversary observing another bid encrypted with a malleable algorithm could construct a more competitive bid without breaking the scheme or learning anything about the other bid.

Many of the public-key encryption schemes we have seen are trivially malleable:

ElGamal: Encryption of m is $(c_1, c_2) = (g^y, m \cdot h^y)$, with $G, g \in G, h$ public and y random. To transform a ciphertext encrypting m into an encryption of $f(m) = 2m$, calculate $(c_1', c_2') = (c_1, 2 \cdot c_2)$.

RSA, Rabin: Semantically secure encryption uses the hardcore bits of the a randomly selected trapdoor permutation f_k with trapdoor t; $pk = k, sk = t$. Encryption chooses $x \overset{\text{R}}{\leftarrow} D_k$ and outputs $c = (f_k(x), b_k(x) \oplus m)$ using the hardcore bits $b_k(x)$. How is this malleable?

"Plain" RSA, Rabin: These are sometimes used directly in practice but they are not semantically secure. Encryption is $c = m^e \bmod n$, with $n = p \cdot q$; n is public and p, q secret. (Rabin encryption is *not* equivalent to RSA with public exponent 2, but this attack applies to Rabin encryption: set $e = 2$.) To transform a ciphertext encrypting m into an encryption of $f(m) = 3m$, calculate $c' = 3^e \cdot c \bmod n \equiv (3 \cdot m)^e \bmod n$.

It is no coincidence that ElGamal, RSA and Rabin encryption are all insecure under chosen ciphertext attack: any malleable encryption scheme allows an adversary to succeed in a chosen ciphertext attack by applying the transformation to the challenge $c = E(m)$. In one informal setting, the adversary computes $c' = f'(c) = E(f(m))$, queries the decryption oracle on c' to recover $f(m)$, then finally inverts $f(m)$ to recover m.

3 Homomorphic Encryption

It turns out that the properties that make encryption schemes malleable have an interesting benefit: parties may *want* to compute valid encryptions of values that are a function of other encrypted values.

Definition 1 *A* homomorphic encryption scheme *is defined by three polynomial-time algorithms (G, E, D), together with at least one pair of polynomial-time computable binary operations $\langle \oplus, \otimes \rangle$ on the plaintext space $\mathcal{P}$ and the ciphertext space $\mathcal{C}$, respectively.*

- *The* key generation *algorithm G is a randomized algorithm that takes a security parameter 1^n as input and returns a pair (pk, sk), where pk is the* public key *and sk is the* secret key*; we write $(pk, sk) \overset{R}{\leftarrow} G(1^n)$.*

- *The* encryption *algorithm E is a stateless randomized algorithm that takes the public key pk and a* plaintext *(aka* message*) m and outputs a ciphertext $c = E_{pk}(m)$. When a random* help value *r is used in the encryption, we write $c = E_{pk}(m, r)$.*

- *The* decryption *algorithm D is a deterministic algorithm (or pair of algorithms) that takes a ciphertext $c = E(m, r)$ and* either *the secret key sk* or *the random help value r (if applicable) and returns a plaintext m. We write $m = D_{sk}(c)$ or $m = D_r(c)$, respectively, depending on its input.*

- *The encryption algorithm E is a "homomorphism" between the plaintext space $\mathcal{P}$ and the ciphertext space $\mathcal{C}$ in the sense that if $c_1 = E(m_1)$ and $c_2 = E(m_2)$, then $c_1 \otimes c_2 \in E(m_1 \oplus m_2)$.*

- *The scheme must also satisfy the property that a verifier who knows a ciphertext c can verify that c is a valid encryption of m, given m and any random help value r in the encryption $c = E(m, r)$.*

We observe that our definition is slightly different from ordinary public-key encryption schemes by the explicit introduction of the random help value used in the probabilistic encryption and its function as a means for decryption.

Exercise (think about this after the end of the lecture): Why don't we consider homomorphic private-key encryption schemes?

3.1 Homomorphic Schemes We Know

Of the public key encryption schemes we have already seen, their homomorphic properties correspond to the malleability weaknesses we described above:

ElGamal: Encryption of m is $(c_1, c_2) = (g^y, m \cdot h^y)$, with $G, g \in G, h$ public and y random. Then given ciphertexts $(g^{y_1}, m_1 \cdot h^{y_1}), (g^{y_2}, m_2 \cdot h^{y_2})$, their pairwise product is $(g^{y_1+y_2}, m_1 \cdot m_2 \cdot h^{y_1+y_2})$, an encryption of $m_1 \cdot m_2$.

"Plain" RSA, Rabin: Encryption is $c = m^e \bmod n$, with $n = p \cdot q$; n is public and p, q secret. Given ciphertexts $c_1 = m_1^e, c_2 = m_2^e$, their product is $(m_1 \cdot m_2)^e \bmod n$, an encryption of $m_1 \cdot m_2$.

We observe that these schemes are multiplicatively homomorphic: $E(a) \cdot E(b) \in E(a \cdot b)$. Certainly this is useful, but it would be nice to have an additively homomorphic scheme – such that $\oplus$ is standard addition and $E(a) \otimes E(b) \in E(a + b)$ for some operation $\otimes$.

4 Paillier Encryption

Paillier's trapdoor function is an isomorphism $f : \mathbb{Z}_N \times \mathbb{Z}_N^* \to \mathbb{Z}_{N^2}^*$ given by $f(a,b) = (1+N)^a \cdot b^N \bmod N^2$, where $N = pq$ for distinct odd primes p, q of equal length. This function f can be efficiently computed but inverting it is believed to be difficult without the factorization of N under the *Composite Residuosity Assumption*.

One can encrypt directly using this trapdoor function by letting a be the message m and b the random help value r. Our scheme is thus defined by three polynomial-time algorithms (G, E, D):

G: Pick two n-bit primes p, q. Set $pk = N = p \cdot q$, $sk = \varphi(N) = (p-1)\cdot(q-1)$.

E: Let $m \in \mathbb{Z}_N$ be the message to encrypt and obtain random help value $r \overset{\text{R}}{\leftarrow} \mathbb{Z}_N^*$. Set $c = E_N(m, r) = (1+N)^m \cdot r^N \equiv (1 + m \cdot N)r^N \bmod N^2$.

D: To decrypt c using $\varphi(N)$, compute $\hat{c} = c^{\varphi(N)} \equiv (1+N)^{m\cdot\varphi(N)} \equiv (1 + m \cdot \varphi(N) \cdot N) \bmod N^2$ (by Fermat's Little Theorem). Then compute $m' = \frac{\hat{c}-1}{\varphi(N)} \bmod N^2$ and recover $m = m'/N$.
(We cannot divide by N modulo N^2 because N, which divides N^2, has no inverse.)
To decrypt c using r, compute $\hat{c} = c \cdot r^{-N} \bmod N^2$, then recover $m = (\hat{c}-1)/N$.
Anyone who knows the secret key sk can recover r from c; $r = c^{N^{-1} \bmod \varphi(N)} \bmod N$.

4.1 The Decisional Composite Residuosity Assumption; Security of Paillier's Scheme

- The DCRA assumption says a random Nth residue is computationally indistinguishable from a random element of $\mathbb{Z}_{N^2}^*$. That is, $(N, R^N) \overset{\text{c}}{\equiv} (N, S)$, where N is a random Paillier modulus, R, R' are random elements of $\mathbb{Z}_N^*$, and S, S' are random elements of $\mathbb{Z}_{N^2}$.

- Then, we observe the following three facts:
  $(N, (1+N)^{m_0} * S) \overset{\text{c}}{\equiv} (N, (1+N)^{m_1} * S')$
  $(pk, E_{pk}(m_0)) \equiv (N, (1+N)^{m_0} * R^N) \overset{\text{c}}{\equiv} (N, (1+N)^{m_0} * S)$
  $(pk, E_{pk}(m_1)) \equiv (N, (1+N)^{m_1} * R'^N) \overset{\text{c}}{\equiv} (N, (1+N)^{m_1} * S')$.

  These imply that $(pk, E_{pk}(m_0)) \equiv (N, (1+N)^{m_0} * R^N) \overset{\text{c}}{\equiv} (N, (1+N)^{m_1} * R'^N) \equiv (pk, E_{pk}(m_1))$, that is, the Paillier scheme is semantically secure under the DCRA.

  This argument uses the general fact that multiplying a fixed element of a group by a uniformly random element gives you a uniformly random element of the group.

- Paillier made a related argument that a successful CPA attacker can break the DCRA. Specifically, assume m_0 and m_1 are two known messages and c is a ciphertext of either m_0 or m_1. $c \in E(m_0)$ if and only if $c \cdot (1+N)^{-m_0} \bmod N^2$ (which the adversary can compute) is an Nth residue. Thus if an adversary has algorithm $\mathcal{A}(c, m_0)$ that can identify whether c is an encryption of m_0 with nonnegligible probability, he can use $\mathcal{A}$ to decide composite residuosity.

4.2 Homomorphic Properties of Paillier Encryption

One of the most attractive properties of the Paillier system is that it is *additively* homomorphic over plaintexts and also allows for multiplication of plaintexts by a constant. All of these primitives can be performed by anyone.

- Addition:

- Multiplication by a constant:

With these primitives, one can divide the plaintext by any constant $k \in \mathbb{Z}_N^*$ (equivalent to multiplying by $k^{-1} \bmod N$). One can subtract two plaintexts via their ciphertexts $c_1 = E(m_1, r_1), c_2 = E(m_2, r_2)$. $c_1 \cdot c_2^{-1} \equiv E(m_1 - m_2, r_1/r_2) \pmod{N^2}$.

Even more complex primitives are possible. Given $c_1 = E(m_1, r_1), c_2 = E(m_2, r_2)$, anyone who has the random help values r_1, r_2 or the secret key *sk* can prove the following facts to a verifier who has only c_1 and c_2, revealing minimal information about m_1 or m_2. (In your homework, you will be asked to show that the Equality proof reveals nothing about m_1 and m_2 by proving that any two pairs of ciphertexts are indistinguishable given only the quotient of their random help values.)

- Equality: Since $m_1 - m_2 = 0$, the prover reveals the single integer $\bar{r} = r_1/r_2 \bmod N$. The verifier verifies that $D_{\bar{r}}(c_1/c_2) = m$ by checking that $(c_1/c_2) \equiv (1+N)^0 \bar{r}^N \equiv \bar{r}^N \pmod{N^2}$.
- Range: $m_1 < 2^t < N$ for constant t (see next subsection)
- Product of Two Plaintexts: $c_3 = E(m_1 \cdot m_2, r_3)$ (see papers cited below)
- Inequality: $m_1 \geq m_2$
  First prove $m_1, m_2 < 2^t$ where t is chosen so $2^t < N/2$. Then compute $c_3 = c_1/c_2 = E(m_1 - m_2, r_1/r_2)$, and prove, using c_3 and the Range primitive, that $(m_1 - m_2) < 2^t < N/2$. This implies $m_1 \geq m_2$, because if not $(m_1 - m_2)$ would wrap around mod N and we could not prove $(m_1 - m_2) < 2^t$. (This necessarily reveals that $m_1, m_2 < 2^t$ but nothing else.)

Some of these primitives and others are explored in detail in a paper "A Generalisation, a Simplification and some Applications of Paillier's Probabilistic Public-Key System" by Dåmgard and Jurik (2001); other formulations are presented in "Practical Secrecy-Preserving, Verifiably Correct and Trustworthy Auctions" by Parkes, Rabin, Shieber and Thorpe (2006).

4.2.1 Proof of Range

(This section is derived from the Parkes et al. paper cited above.) Given ciphertext $c = E(m, r)$ we need to prove that $m < 2^t$ for some constant t.

Definition 2 *A valid* test set *S for the assertion "$c = E(m, r)$ is an encryption of a number $x < 2^t$" is a set of $2t$ randomly ordered encryptions, $S = \{G_1 = E(u_1, s_1), \ldots, G_{2t} = E(u_{2t}, s_{2t})\}$, where each of the powers of 2 – $\{1, 2, \ldots, 2^{t-1}\}$ – appears among the u_i exactly once and the remaining t values u_j are all 0.*

By use of such a test set S, the prover can prove that $x < 2^t$ as follows:

Let $m = 2^{t_1} + \ldots + 2^{t_\ell}$ be the representation of m, a sum of distinct powers of 2. The prover selects ℓ randomly ordered encryptions $G_{j_1}, \ldots, G_{j_\ell}$ of $2^{t_1}, \ldots, 2^{t_\ell}$, and further $t - \ell$ encryptions $G_{j_{\ell+1}}, \ldots, G_{j_t}$ of 0.

The prover hands over these t encrypted "bits", $\{G_{j_1}, \ldots, G_{j_t}\}$, to the verifier. The prover also hands over the random help value that proves that the sum of the bits is equal to m, i.e., the original random help value r divided (mod N^2) by the product of all the random help values in these t bits' encryptions: $\bar{r} = r/(\prod_{i=1}^{t} s_{j_i})$.

The verifier can now calculate the sum of the "bits" m' by computing the product of the encryptions: $c' = E(m', s) = \prod_{i=1}^{t} G_{j_i}$ and verifying that $m' = m$ by calculating and checking $c/c' \equiv E(m - m', r/s) \equiv \bar{r}^N (\text{mod } N^2)$.

This reveals nothing to the verifier because she cannot tell whether particular bits represent 0 or 1, and the order given to her is random. However, there is a slight problem in that the verifier cannot be sure that the test set containing the $2t$ encryptions of 0 and 1 is well-formed.

To get around this, we employ what is called a "cut and choose" protocol. The prover creates a large number $2k$ of sample test sets; the verifier then asks the prover to "unlock" half (k) of them by revealing the random help values used to encrypt each element in each test set. The verifier checks that each test set contains exactly t encryptions of 0 and t encryptions of the powers of 2 from $2^0 \ldots 2^{t-1}$.

The only way that the prover can cheat is if the k test sets that the verifier chose to be revealed were valid and the k remaining ones were invalid. The probability of such an unfortuitous choice is $\binom{2k}{k}^{-1}$. For $k = 20$, that probability is about $\sqrt{\frac{20\pi}{2^{40}}} < \frac{8}{10^{12}}$ by Sterling's Theorem.

5 Applications of Homomorphic Cryptography

5.1 Auctions

Given the primitives we have above, we can construct a simple verifiable sealed-bid auction protocol among n bidders $P_1, \ldots, P_n$ for a single item as follows:

1. An auctioneer is chosen with a Paillier key pair pk, sk as above
2. Each bidder P_i secretly encrypts the price v_i she wishes to pay $\hat{v}_i = E_{pk}(v_i, r_i)$ and publishes $\hat{v}_i$ to everyone. Only the auctioneer can see what her bid is.
3. When the auction closes, the auctioneer secretly decrypts all the bids and identifies the winner (without loss of generality) P_1.
4. The auctioneer uses our Inequality primitive above to prove, using only $\hat{v}_1$ and $\hat{v}_j$, that $v_1 > v_j$ for $2 \leq j \leq N$.
5. The auctioneer, if he wishes, may reveal the winning bid by revealing the value r_1.
6. The auctioneer can even reveal a Vickrey price by revealing the second price value r_2. In more complex auctions, the auctioneer can even prove correct optimal prices that are arbitrary linear functions of the encrypted bid values!

5.2 Voting

Paillier is an attractive protocol for voting because it is additively homomorphic, and makes it possible to count votes without decrypting any voter's particular vote.

1. The election board creates a distributed Paillier key pair (pk, sk) so that no one entity knows the secret decryption key but the public encryption key can be published.
2. A constant k is chosen large enough so no more than 2^k votes will be cast. Each candidate is associated with a value $C_i = 2^{ik}$. For example, $k = 32$ (about 4 billion votes), and Adams $= 2^{32}$, Buchanan $= 2^{64}$, Coolidge $= 2^{96}$.
3. When a voter votes, their vote is an encryption of $1 + C_i$. For example, a vote for Buchanan would be $V_j = (1 + 2^{64})$, encrypted: $\hat{V_j} \in E_{pk}(1 + 2^{64}, r_j)$. The voting machine computes these values.
4. The voting machine proves that a vote is correct without revealing any information about the vote, using a protocol similar to the primitive that proves a value $m < 2^t$. Amazingly, the machine can do this without knowing the decryption key – provided it knows the random help value used to encrypt.
5. The votes are tallied by computing the product of the encrypted votes: $\prod_j \hat{V_j} \in E_{pk}(\sum_j V_j)$.
6. The raw votes are destroyed. The election officials then reconstruct the secret decryption key and decrypt the total. By taking the total modulo each successive C_i, the election board can extract the number of votes for each candidate.

5.3 What other applications can we think of?

- Multi-Party Computation, e.g. the Professors' Salary Problem
-
-

5.4 Some Security Considerations

Clearly, "plain" Paillier is not appropriate as the only mechanism in an auction protocol. (Why?)

We have not yet studied digital signatures and message authentication codes; later we will see how they complement "plain" homomorphic encryption schemes to prevent a participant in a protocol from creating an encryption based on someone else's published encrypted values.

5.5 Other Facts About Homomorphic Encryption

It remains an important open question whether one can construct a homomorphic encryption scheme that is *doubly* homomorphic in the ciphertexts, that is, there exist two pairs of operations $\langle +, \oplus \rangle$, $\langle \times, \otimes \rangle$ such that $E(a+b) = E(a) \oplus E(b)$ and $E(a \times b) = E(a) \otimes E(b)$. Paillier's scheme is sometimes said to be doubly homomorphic, but its multiplicative homomorphism is limited to creating new encryptions of products of plaintexts with constant factors. If someone were to construct a truly doubly homomorphic scheme allowing both addition and multiplication over plaintexts by operations on the ciphertexts, what would the implications be?