CPSC 467b: Cryptography and Computer Security

Instructor: Michael Fischer
Lecture by Ewa Syta

Lecture 21
April 4, 2012

Encryption with Special Properties

Goals of encryption

The main goal of encryption is to provide data confidentiality.

Normally, there is a lot you do with your unencrypted data: analyze, search, compute, etc.
However, once data is encrypted there is not much you can do with it.

Encrypted data $\longrightarrow$ secured and useless
Unencrypted data $\longrightarrow$ unsecured and useful [1]

[1] Slight oversimplification

Working with encrypted data

Solution: Encrypt – decrypt – perform operations – re-encrypt

Problems: Can get very expensive very quickly. Privacy issues.

Another solution: Perform at least *some* operations on encrypted data *without* decrypting it.

Problems: How do we do that? What operations should be allowed? Will it affect security properties of the encryption scheme?

Encryption with special properties

The goal is to design an encryption function E so that we can perform meaningful operations on the ciphertexts without decrypting it.

To make it possible, E would have to "give" some special properties to the ciphertext.

Homomorphic Encryption

Homomorphic encryption

Informally, homomorphic encryption is an encryption scheme with a special property that allows operations applied to ciphertext be preserved and carried over to the plaintext.

Group homomorphism

A *group homomorphism* from $(G, *)$ to $(H, \cdot)$ is a function $h : G \longrightarrow H$ such that $\forall u, v \in G$ it holds that $h(u * v) = h(u) \cdot h(v)$.

Homomorphic encryption

Let $\mathcal{M}$ be the set of plaintext messages, $\mathcal{C}$ be the set of ciphertext messages, and $\mathcal{K}$ be the set of keys.

An encryption scheme is *homomorphic* if for any given encryption key $k \in \mathcal{K}$ the encryption function E satisfies:

$$\forall m_1, m_2 \in \mathcal{M},\ E(m_1 \odot_{\mathcal{M}} m_2) = E(m_1) \odot_{\mathcal{C}} E(m_2)$$

for some operators $\odot_{\mathcal{M}}$ in $\mathcal{M}$ and $\odot_{\mathcal{C}}$ in $\mathcal{C}$,

C. Fontaine and F. Galand, *A Survey of Homomorphic Encryption for Nonspecialists*, EURASIP Journal on Information Security, 2007

Group homomorphism

We want $(\mathcal{M}, \odot_{\mathcal{M}})$ and $(\mathcal{C}, \odot_{\mathcal{C}})$ to form a group homomorphism under E.

Types of homomorphism

An encryption scheme can be homomorphic with respect to one or more group operators.

An encryption scheme is *additively* homomorphic if we consider the addition operator, and *multiplicatively* homomorphic if we consider the multiplication operator.

Types of homomorphic encryption

Partially homomorphic encryption – it is possible to perform operations on encrypted data with respect to one group operator. For example, from $E(x)$, $E(y)$ compute $E(x + y)$ but not $E(x * y)$.

Fully homomorphic encryption – it is possible to perform operations on encrypted data with respect to two group operator. For example, from $E(x)$, $E(y)$ compute $E(x + y)$ and $E(x * y)$.

Somewhat homomorphic encryption – it is possible to perform a limited number of operations on encrypted data with respect to two group operator. For example, we can perform only a few additions and multiplications before the the scheme fails.

Applications of homomorphic encryption

- Cloud computing (untrusted third parties can be used)
- E-voting (votes can be counted without revealing what they are)
- Private information retrieval (searching encrypted databases)

Partially homomorphic encryption schemes

There is a number of encryption schemes which have the desired homomorphic property.

We will have a look at the following schemes:

- RSA
- ElGamal
- Goldwasser-Micali

(Plain) RSA

Public key: (e, N)
Private key: (d, N)
Encryption function: $E(m) = m^e \bmod N$

Multiplicatively homomorphic property:

$$E(m_1) * E(m_2) =$$
$$m_1^e * m_2^e \bmod N =$$
$$(m_1 * m_2)^e \bmod N =$$
$$E(m_1 * m_2)$$

ElGamal

Public key: (p, g, b), where $b = g^x$
Private key: (x)
Encryption function: $E(m) = (g^r, m * b^r)$ for a random $r \in Z_{\phi(p)}$

Multiplicatively homomorphic property:

$$\begin{aligned} E(m_1) * E(m_2) = \\ (g^{r_1}, m_1 * b^{r_1})(g^{r_2}, m_2 * b^{r_2}) = \\ (g^{r_1+r_2}, (m_1 * m_2)b^{r_1+r_2}) = \\ E(m_1 * m_2) \end{aligned}$$

Goldwasser-Micali

First provably secure randomized encryption scheme.
Based on the intractability of the quadratic residuosity problem modulo composite N.
Very inefficient: a single bit after encryption is approximately $|N|$!

Public key: (x, N), where x is a quadratic non-residue mod N
Private key: (p, q), where $N = p * q$
Encryption function: $E(b) = r^2 x^b \bmod N$, where b is one bit of plaintext and $r \in Z^*_{\phi(N)}$
Decryption function: If $E(b) \in QR_N$, then $b = 0$, otherwise $b = 1$.

Goldwasser-Micali

Encryption function: $E(b) = r^2 x^b \bmod N$

Additively homomorphic property:

$$\begin{aligned} E(b_1) * E(b_2) = \\ r_1^2 x^{b_1} r_2^2 x^{b_2} = \\ (r_1 r_2)^2 x^{b_1 + b_2} = \\ E(b_1 \oplus b_2) \end{aligned}$$

Fully homomorphic encryption

The first fully homomorphic encryption scheme using lattice–based cryptography was presented by Craig Gentry in 2009.[2]

Later in 2009 a second fully homomorphic encryption scheme which does not require ideal lattices was presented.[3]

[2] C. Gentry, *Fully Homomorphic Encryption Using Ideal Lattices* ACM Symposium on Theory of Computing (STOC), 2009

[3] M. van Dijk, C. Gentry, S. Halevi and V. Vaikuntanathan *Fully Homomorphic Encryption over the Integers*, Eurocrypt 2010

FHE performance

Gentry estimated[4] that performing a Google search with encrypted keywords would increase the amount of computing time by about a trillion. Moore's law calculates that it would be 40 years before that homomorphic search would be as efficient as a search today.

At Eurocrypt 2010, Craig Gentry and Shai Halevi presented a working implementation of fully homomorphic encryption.
Martin van Dijk about the efficiency:

"Computation, ciphertext-expansion are polynomial, but a rather large one..."

[4] M. Cooney, *IBM Touts Encryption Innovation. New technology performs calculations on encrypted data without decrypting it* computerworld.com

Security of homomorphic encryption

Let's (informally) rephrase what homomorphic encryption is.

"If you encrypt some plaintext using homomorphic encryption, then by changing ciphertext you can change the corresponding plaintext".

Is it a good or bad property?

Security notions for encryption schemes

Combination of security goals and attack scenarios.
Goals:

- Indistinguishability (IND). The attacker does not learn anything about plaintext x from ciphertext y.
- Non-malleability (NM). Based on ciphertext y the attacker cannot produce y' so that the corresponding plaintexts x and x' are meaningfully related.

Attack scenarios:

- Chosen–plaintext attack (CPA).
- Non–adaptive chosen–ciphertext attack (CCA1).
- Adaptive chosen–ciphertext attack (CCA2).

Security of homomorphic encryption

Non–malleability is a desirable security goal for encryption schemes so that the attacker cannot tamper with the ciphertext to affect the plaintext and go undetected.

However, homomorphic encryption implies malleability!

No homomorphic encryption scheme can be secure against IND-CCA. Alternatively: the highest security level a homomorphic scheme can achieve is IND-CPA.

More on homomorphic security

To reconcile this situation, we want an encryption scheme to be non-malleable except for some desired operations.

It's difficult to capture the notion of "some malleability allowed."

Benignly-malleable (gCCA) was proposed as a relaxation of CCA security and was further relaxed in the definition of Replayable-CCA (RCCA) security. RCCA security allows a scheme to have homomorphic operations which preserve the underlying plaintext, but enforces non-malleability "everywhere else".[5]

[5] For more on security of homomorphic encryption see B. Hemenway and R. Ostrovsky, *On Homomorphic Encryption and Chosen-Ciphertext Security*, Proceedings of PKC 2012

Question

Can you think of a symmetric homomorphic encryption scheme?

Encryption with Other Properties

Encryption schemes

We will have a look at the following schemes:

1. Searchable encryption
2. Deniable encryption
3. Signcryption
4. Identity-based encryption

Searchable encryption

Searchable encryption[6] allows to test whether certain keywords are included in a ciphertext message without decrypting it or learning anything about its content.

[6] D. Boneh, G. Crescenzo, R. Ostrovsky and G. Persiano, *Public Key Encryption with Keyword Search*, Advances in Cryptology – EUROCRYPT 2004

SES scenario

Suppose Alice wishes to read her email on a number of devices. Alices mail gateway can route emails based on the keywords in the email. For example, when Bob sends email with the keyword "urgent" the email is routed to Alices pager.

Bob sends encrypted email to Alice using Alice's public key. Both the contents of the email and the keywords are encrypted.

The goal is to enable Alice to give the gateway the ability to test whether "urgent" is a keyword in the email, but the gateway should learn nothing else about the email.

SES details

1. Bob encrypts his email using a standard public key system.
2. Then he appends to the resulting ciphertext a Searchable Encryption (SES) of each keyword.
3. To send a message M with keywords $W_1, \ldots W_m$ Bob sends

$$E_{A_{pub}}(M)||SES(A_{pub}, W_1)|| \ldots ||SES(A_{pub}, W_m)$$

where A_{pub} is Alices public key.

SES details cont.

5. Alice gives the gateway a certain trapdoor T_W that enables the gateway to test whether one of the keywords associated with the message is equal to the word W.

   Given $SES(A_{pub}, W')$ and T_W the gateway can test whether $W = W'$.

   If $W \neq W'$ the gateway learns nothing more about W'.

   Alice and Bob do not need to communicate as Bob generates the searchable encryption for W' just given Alice's public key.

SES a bit more formally

A SES scheme consists of the following polynomial time randomized algorithms:

1. $KeyGen(s)$: takes a security parameter s and generates a public/private key pair (A_{pub}, A_{priv}).
2. $SES(A_{pub}, W)$: for a public key A_{pub} and a word W produces a searchable ciphertext of W.
3. $Trapdoor(A_{priv}, W)$: given Alice's private key and a word W produces a trapdoor T_W.
4. $Test(A_{pub}, S, T_W)$: given Alice's public key, a searchable encryption $S = SES(A_{pub}, W')$, and a trapdoor $T_W = Trapdoor(A_{priv}, W)$, outputs "yes" if $W = W'$ and "no" otherwise.

Deniable encryption

Deniable encryption[7] allows an encrypted message to be decrypted to a different plausibly looking plaintexts, depending on the input information used.

This feature give the sender *plausible deniability* if compelled to reveal the encryption information.

[7] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, *Deniable Encryption*, Proceedings of Crypto 1997

Deniable encryption scenario

Regular encryption schemes provide confidentiality of encrypted data in the presence of a (powerful) adversary who given a ciphertext is trying to learn the corresponding plaintext.

However, assume that Eve is evil and has a lot of power over Alice and Bob who wish to communicate. Specifically, Eve can approach Alice or Bob after they exchanged an encrypted message and demand all private information: the plaintext, the randomness used for encryption, and the keys. Once Eve obtains this information, she can verify that it matches the transmitted ciphertext since the encryption function is known.

It is difficult to provide security in such attack scenario. Amazingly, deniable encryption offers some protection against this very different and more hostile attack.

Types of deniable encryption schemes

Deniable encryption schemes can be categorized according to which parties may be coerced:

- Sender deniable
- Receiver deniable
- Sender–and–receiver deniable

Deniable encryption can be symmetric or asymmetric.

Public-key sender-deniable encryption scheme

Assume that only the receiver possesses the decryption key, m is the correct plaintext and $c = E(m, r)$ is the corresponding ciphertext where r is the key and possibly other input parameter.

If approached by Eve, Alice can reveal fake parameters like random strings and private keys that yield a plaintext m_f instead of the original plaintext m.

The goal is to present r_f and $m_f \neq m$, such that $c = E(m_f, r_f)$ where m_f is a plausibly looking plaintext. The protocol for finding such m_f and r_f is called a *faking algorithm*.

Shared–key receiver–deniable ElGamal encryption[8]

Preliminaries: Alice and Bob has a shared secret s. Bob's public key is (p, g, y). Bob's private key x is known to Alice.

"Normal" encryption: Alice sends (α, β), where $\alpha = g^k$ and $\beta = m \cdot y^k$, where k is a randomly chosen value.

Deniable encryption: To encrypt a fake message m_f and an "illegal" message m, Alice computes $k = HASH(s||m_f)$. Then, she computes $\alpha = g^k \cdot m$, $\beta = (y^k \cdot m^x) \cdot m_f$.

The (α, β) is a regular ElGamal ciphertext of m_f.

[8] M. Klonowski, P. Kubiak, and M. Kutylowski, *Practical Deniable Encryption*, Proceedings of SOFSEM, 2008

Shared–key receiver–deniable ElGamal encryption

"Normal" encryption:

$$\alpha = g^k \text{ and } \beta = m \cdot y^k$$

Deniable encryption:

$$\alpha = g^k \cdot m \text{ and } \beta = m_f \cdot (y^k \cdot m^x)$$

Message m is in fact sent subliminally – a covert channel is created.

Shared–key receiver–deniable ElGamal encryption

"Normal" decryption:

$$\beta \cdot \alpha^{-x} = (y^k \cdot m) \cdot g^{-kx} = m$$

Decryption:
Bob needs to retrieve the "legal" message m_f:

$$\beta \cdot \alpha^{-x} = m_f \cdot (y^k \cdot m^x) \cdot (g^k \cdot m)^{-x} = m_f.$$

Then he computes $k = HASH(s||m_f)$ and $m = \alpha \cdot g^{-k}$.

Dishonest opening: Bob, if coerced, can reveal his key x. The coercer can check that (α, β) is in fact a regular, valid ElGamal encryption of the message m_f. Therefore, Bob is able to mimic decryption of the fake message m_f.

Shared–key receiver–deniable ElGamal encryption

This scheme provides perfect receiver deniability: the transcript of sending m is indistinguishable from sending m_f.

The scheme is not sender–deniable: Alice has no effective algorithm that for an argument $\alpha = g^k \cdot m$ returns an exponent k' s. t. $\alpha = g^{k'}$. Why?

Also, the fact that Alice knows x is not desirable. Why?

Using a well known and widely used (as opposed to a new, designed for this purpose) scheme improves "deniability".

Applications of deniable encryption

Prevention of vote buying in electronic voting. A coercer may offer a bribe in exchange for proof of a person's vote after seeing the corresponding encrypted vote.

Storing encrypted data in a deniable way.

Incoercible multiparty computation. Participants are able to keep their inputs private even in the presence of a coercer.

Signcryption

Encryption and signature schemes are the basic tools offered by public key cryptography.

They are normally viewed as important but distinct building blocks for higher level protocols, but there are many settings were both are needed.

Signcryption [9] is a scheme that provides both functionalities simultaneously.

[9] From ECRYPT report D.AZTEC.7, *New Technical Trends in Asymmetric Cryptography*.

Signcryption scenario

Consider a secure email as an example.

Encryption provides confidentiality of the content and digital signature provides authentication.

Performing both operations at once improves efficiency and usability.

Signcryption more formally

A signcryption scheme S consists of the following algorithms.

1. Sender and receiver key generation algorithms K_s and K_r take as input a security parameter and return a matching public/secret key pair (pk_s, sk_s) for the sender and (pk_r, sk_r) for the receiver.
2. The randomized signcryption algorithm S takes as input a sender's keys (pk_s, sk_s), a receiver's public key pk_r and a plaintext m and returns a ciphertext σ.
3. The deterministic unsigncryption algorithm U takes as input a sender's public key pk_s, a receiver's keys (pk_r, sk_r), and a string σ and returns either a message m or the symbol $\perp$ if the signcryption was invalid.

Signcryption = encryption + digital signature?

Not exactly. Unlike a "standalone" digital signature scheme, a signcryption does not support non-repudiation of messages by default. Why?

Signcryption constructions

A number of signcryption schemes has been proposed.

They all require an encryption scheme that is IND-CCA2 secure and a digital signature scheme that is EUF-CMA (existentially unforgeable under adaptive chosen message attack).

Identity-based encryption

ID-based encryption allows to use some known aspect of the user identity, for example an email address or IP address, to generate a public key.

This means that Alice can send confidential messages to anyone, even people who has not set up their public keys yet!
Unfortunately, it will be a bit tricky to retrieve the corresponding private key.

First was proposed by Adi Shamir in 1984 as one solution to the key distribution problem.[10]

[10] Adi Shamir, Identity-Based Cryptosystems and Signature Schemes. Advances in Cryptology: Proceedings of CRYPTO 84

Identity-based encryption system

Public keys are derived from a known identity value such as an arbitrary ASCII string. Corresponding private keys are generated by a trusted third party called the Private Key Generator (PKG).

Before ID-based system can be used, a PKG needs to be established and its master public key made available.

If Bob wants to send an email to Alice, he computes her public key by combining the master public key with Alice's identity information.

If Alice wants to read Bob's message, she contacts PKG, which uses the master private key to generate the private key for Alice.

Issues with Identity-based encryption

There three big issues with this scheme:

1. PKG must be trusted
2. Authenticity of the private key requestor needs to be established.
3. Private key must be securely transmitted.

Additional Resources

More information on encryption with special properties:
New Technical Trends in Asymmetric Cryptography, ECRYPT report D.AZTEC.7,

`http://www.ecrypt.eu.org/ecrypt1/documents/D.AZTEC.7.pdf`

Tutorial on homomorphic encryption by Shai Halevi presented at Crypto 2011. Video and slides.

`http://people.csail.mit.edu/shaih/presentations.html`