CPSC 467: Cryptography and Computer Security

Michael J. Fischer

Lecture 25
December 6, 2017

Encryption with Special Properties

Goals of encryption

The main goal of encryption is to provide data confidentiality.

Normally, there is a lot you can 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 $\rightarrow$ secured and useless
Unencrypted data $\rightarrow$ 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

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.

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 operators. For example, we can only evaluate low-degree polynomials over encrypted data.

Applications of homomorphic encryption

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 are many encryption schemes which have the desired homomorphic property.

You should be familiar with at least some of them:

- RSA (multiplicatively)
- ElGamal (multiplicatively)
- Goldwasser-Micali (additively)

(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:

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

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]

A lot of changes since then.

[2] C. Gentry, *Fully Homomorphic Encryption Using Ideal Lattices*, 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] *IBM Touts Encryption Innovation. New technology performs calculations on encrypted data without decrypting it* computerworld.com, M. Cooney,

Current FHE efforts

FHE is a very popular research area. Three main directions:

1. Improving the scheme itself (security)
   - Relying on standard hardness assumptions
2. Improving the bootstrapping phase (efficiency)
   - Leveled FHE schemes that are initialized with a bound on the maximal evaluation depth
3. Implementations
   - HElib[5], a software library that implements homomorphic encryption.
   - Extension of HElib that includes the full bootstrapping phase[6]

[5] GitHub Repository

[6] S. Halevi and V. Shoup. *Bootstrapping HElib*, 2014

Security of homomorphic encryption

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

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

Q: Is it a good or bad property?

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!

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

However, it's difficult to capture the notion of "some malleability allowed."[7]

[7] B. Hemenway and R. Ostrovsky, *On Homomorphic Encryption and Chosen-Ciphertext Security*, PKC 2012

Encryption schemes

Examples of encryption schemes with special properties:

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

Searchable encryption

Searchable encryption[8] allows to test whether certain keywords are included in a ciphertext message without decrypting it or learning anything about its content.

Scenario: Alice wishes to read her email on different devices. She wants her mail server can route emails based on the keywords in the email. For example, Bob's email labeled "urgent" goes to her phone.

[8] D. Boneh, G. Crescenzo, R. Ostrovsky and G. Persiano, *Public Key Encryption with Keyword Search*, EUROCRYPT 2004

SES highlights

Bob encrypts his email using a standard public key system and Alice's standard public key A_{pub}, and appends to the resulting ciphertext a Searchable Encryption (SES) of some keywords $W_1, \ldots, W_n$.

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

Alice gives the server a trapdoor T_W which depends on A_{priv} and a keyword W. This enables the server to test whether the SES ciphertext equals to some known keyword(s) but nothing else.

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.

Signcryption scenario: encrypting and signing at the same time improves efficiency and usability.

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

Identity-based encryption[10]

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.

Suddenly Alice can send confidential messages to anyone, even people who has not set up their public keys yet!

Tricky to retrieve the corresponding private key: trusted Private Key Generator (PKG), authenticating the key owner, secure transmission of the private key, etc.

[10] Adi Shamir, *Identity-Based Cryptosystems and Signature Schemes*, CRYPTO 1984

Deniable encryption

Deniable encryption[11] allows an encrypted message to be decrypted to different plausibly looking plaintexts, depending on the input information used.

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

[11] R. Canetti, C. Dwork, M. Naor, and R. Ostrovsky, *Deniable Encryption*, 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 custom agents at border crossings have the authority to request decryption keys to encrypted data on one's laptop. Let Chuck be a custom agent.

It is difficult to provide security in such an 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.

Shared–key receiver–deniable ElGamal encryption[12]

Alice and Bob have a shared secret s. Bob's public key is (p, g, y). Alice knows Bob's private key x. Let m_f be a fake message and m_s be a secret message to encrypt deniably.

Normal encryption: Alice computes $\alpha = g^k$ and $\beta = m \cdot y^k$, where k is a randomly chosen value.

Deniable encryption: Alice computes: $\alpha = g^k \cdot m_s$,
$\beta = m_f \cdot (y^k \cdot m_s^x)$, where $k = HASH(s||m_f)$,

The (α, β) pair looks like a regular ElGamal ciphertext of m_f.

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

Shared–key receiver–deniable ElGamal encryption

Normal ElGamal encryption:

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

Deniable ElGamal encryption:

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

Message m_s is in fact sent subliminally – a covert channel is created. Think "nested dolls" encryption.

Shared–key receiver–deniable ElGamal encryption

Normal decryption:

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

Deniable decryption: Bob needs to retrieve the fake message m_f:

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

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

Dishonest opening: Bob, if asked by Chuck, can reveal his key x. Chuck can check that (α, β) is a valid ElGamal encryption of 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 $\alpha = g^k \cdot m_s$ 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".

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`

Bitcoins

Bitcoins

Bitcoins are a kind of virtual digital currency based on cryptography.

- They exist only in the cloud.
- Their supply is limited.
- Like commodities, their value goes up and down depending on market forces.
- They can be used for online transactions without involving a central party.
- Bitcoin transactions are (in some circumstances) anonymous and are a favored medium of exchange for illegal transactions.

Bitcoin exchange

Bitcoins are bought and sold on exchanges such as CEX·IO, where today's bitcoin price is $412 USD. Other hits on a Google search for "bitcoin exchange" are coinbase, CoinDesk, and BTCe.

Bitcoins are volatile. Here's an hour-by-hour graph for the last 24 hours taken from the BTCd site:

Bitcoins in the news

Bitcoins are frequently in the news. Depending on who is talking, they:

- Are the transaction medium of the future.
- Are a threat to national security.
- Are of use primarily to drug dealers and criminals.
- Are a good investment.
- Are the analog of cash on the internet.
- Are secure because they don't rely on trust in a government.
- Are a kind of Ponzi scheme that will sooner or later collapse.

How to understand bitcoins

To evaluate these claims, one must first understand how they work.

- Physically, a bitcoin is an entry in a distributed database called the *block chain*.
- The database is a linked list of transaction *blocks*, each containing a set of transactions.
- A bitcoin is identified by an *address*, which is a public cryptographic key. The owner holds the corresponding private key in her *bitcoin wallet*.
- From a user's perspective, bitcoin is like a mobile app that provides a personal bitcoin wallet.

Bitcoin database

- The database is created and managed by a large number of *miners* operating in a consensus network.
- The consensus network is claimed to be the first decentralized peer-to-peer payment system that is powered by its users with no central authority or middlemen.

Bitcoin transactions

- A transaction takes one or more bitcoins as inputs and produces one or more new bitcoins as outputs.
- The total value of the input bitcoins equals the total value of the output bitcoins.
- The transaction specifies the address(es) at which the output bitcoin(s) are stored.
- The parties involved in a bitcoin transaction work together to create the transaction record. The owners of the input bitcoins must all sign the transaction. The receiving parties must create addresses for the new coins.

Committing a transaction

Once a transaction record has been created, it must be entered into the database.

- The transaction creators broadcast the transaction to all miners.
- Each miner enters the transaction into a list of pending transactions.
- Every now and then, a miner incorporates all valid pending transactions into a new block and adds the new block to the block chain.
- The new block is broadcast to all other miners, who then add it to their copies of the old block chain.

Reaching consensus

A transaction can only be committed to a new block if the source bitcoins have not already been spent.

This is how bitcoins cope with the problem of double spending.

Because of decentralization, it is possible for the same bitcoins to be used in multiple transactions and those conflicting transactions sent to multiple miners.

It is also possible for two miners to extend the block chain in inconsistent ways.

Bitcoin contains a number of mechanisms designed to make such conflicts rare events and to allow for consensus to be reached in case they co occur.

Puzzles

To add a new block to the block chain, a miner must first solve a time-consuming puzzle.

The difficulty of the puzzles is adjusted dynamically so that among all the miners, it takes approximately 10 minutes for the puzzle to be solved.

The first miner to solve the puzzle extends the block chain and sends the new block to the other miners.

A miner who receives a certified block chain that is longer than the one he is currently working on discards is current chain and accepts the new one.

In this way, most miners agree most of the time on what is the current block chain.

SHA-256 puzzles

The puzzle that bitcoin uses is based on the SHA-256 hash function.

Each block chain has an associated hash value.

The puzzle is to find a nonce which, when hashed together with the old hash value, yields an output that begins with a specified number of 0's. The more 0's required, the harder the problem.

Bitcoin economics

Whenever a miner solves a puzzle and extends the block chain, a new bitcoin is created as a bonus and inserted into the block at the miner's address.

- The size of the bonus is algorithmically determined.
- It decreases over time and eventually goes to zero.
- These bonuses are a major incentive for the miners.
- As bonuses get smaller and mining gets more expensive, miners will have to turn to transaction fees to pay their costs.

When is my transaction fully committed?

The answer is, "never".

A dishonest miner can go back in time to an earlier block and try to extend the block chain in a new direction, perhaps one that alters or excludes an earlier transaction.

To do so, he must solve a sequence of puzzles before any other miner solves the "current" puzzle.

The further back in time, the more unlikely it is that the dishonest miner can succeed.

So a given transaction becomes more and more secure as more and more new blocks are successfully added to the block chain.

Trust

Trust depends on having a large number of independent miners, no small subset of which has a majority of the computational power.

In the beginning, the bonus reward structure provided the incentives for many miners to enter the competition.

As mining becomes more expensive and less lucrative, one can expect the number of miners to decrease, possibly leading to a total collapse of the system.

Anonymity

Bitcoin transactions are often said to be anonymous.

It's true that they contain only a public key, not the user's personal information.

However, if the key can be linked to an individual, the anonymity is lost.

There are many ways for such linking to take place, e.g., by confiscating the user's wallet, or by monitoring a user's transaction in progress.

Even if the underlying transaction protocol is anonymous, users of a bitcoin exchange will likely be identified by the exchange in order to enable the conversion of bitcoins to or from conventional currencies.

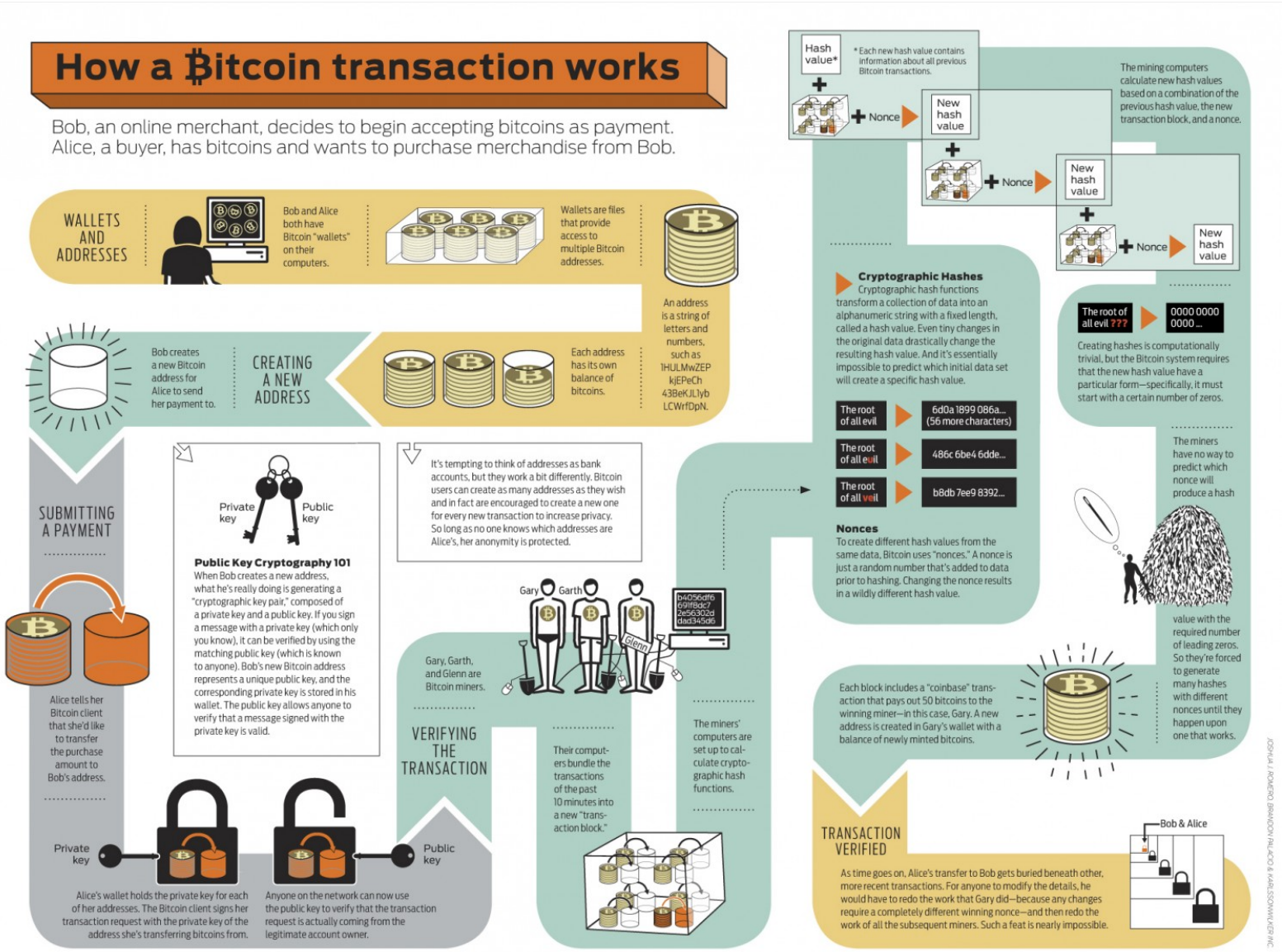
Summary of the transfer protocol

Here's how Alice transfers a bitcoin to Bob:

1. Alice creates and signs a transaction request giving the coin to Bob.
2. The transaction is then broadcast to all of the miners.
3. Each miner first verifies the validity of the transaction by using its current most-recent copy of the database.
4. If valid, the miner attempts to create a new certified database incorporating the new transaction (along possibly with others) into the current database.
5. To certify a database requires solving a computationally-intensive puzzle.

Outline of the transfer protocol (cont.)

6. The puzzle consists of finding a nonce y such that the SHA-256 hash of the database together with y yields a hash value beginning with a long string of 0's.
7. A successful miner broadcasts the new database to all other miners.
8. Each miner upon receiving a new certified database discards all older ones and begins working with the newer one.
9. The system never reaches consensus, but the probability of a certified database being discarded in favor of another decreases exponentially over time.



From http://lsvp.com/2013/03/28/how-bitcoin-works/

Analysis

Why is consensus almost-certainly reached?

- Suppose two miners solve a puzzle simultaneously.
- Both broadcast their versions of the new database D and D'.
- Perhaps half of the miners work on D and half on D'.
- Most miners are likely attempting to incorporate Alice's transaction into a new database.
- Suppose some miner working on D solves the puzzle and sends out the new database D''.
- All miners receiving D'' discard the old D or D' and begin working on D''.
- Now an overwhelming majority of them believe D'' is the current database. They will only change their minds if a yet-longer certified database shows up.

Where's my money?

A good question to ask is, "Where is my money?".

It's obviously in the cloud, but where it is exactly is in the miners' computers.

Security relies on there being many honest miners.

Successful miners are currently rewarded with new bitcoins, but as time goes on, the rewards are programmed to diminish.

What happens when miners no longer have the incentive to solve the computationally-intensive puzzles?

Other potential problems

There are other potential problems as well.

- What happens if Alice's private signing key gets compromised?
- What happens to bitcoins that are lost?
- What happens if the puzzle turns out to be not as hard as expected?
- What happens if people turn their attention to a competing digital cash scheme?
- Is this another Ponzi scheme? Why or why not?
- Bitcoins have been compared to gold. Is that comparison valid?