Broadcast encryption: introduction

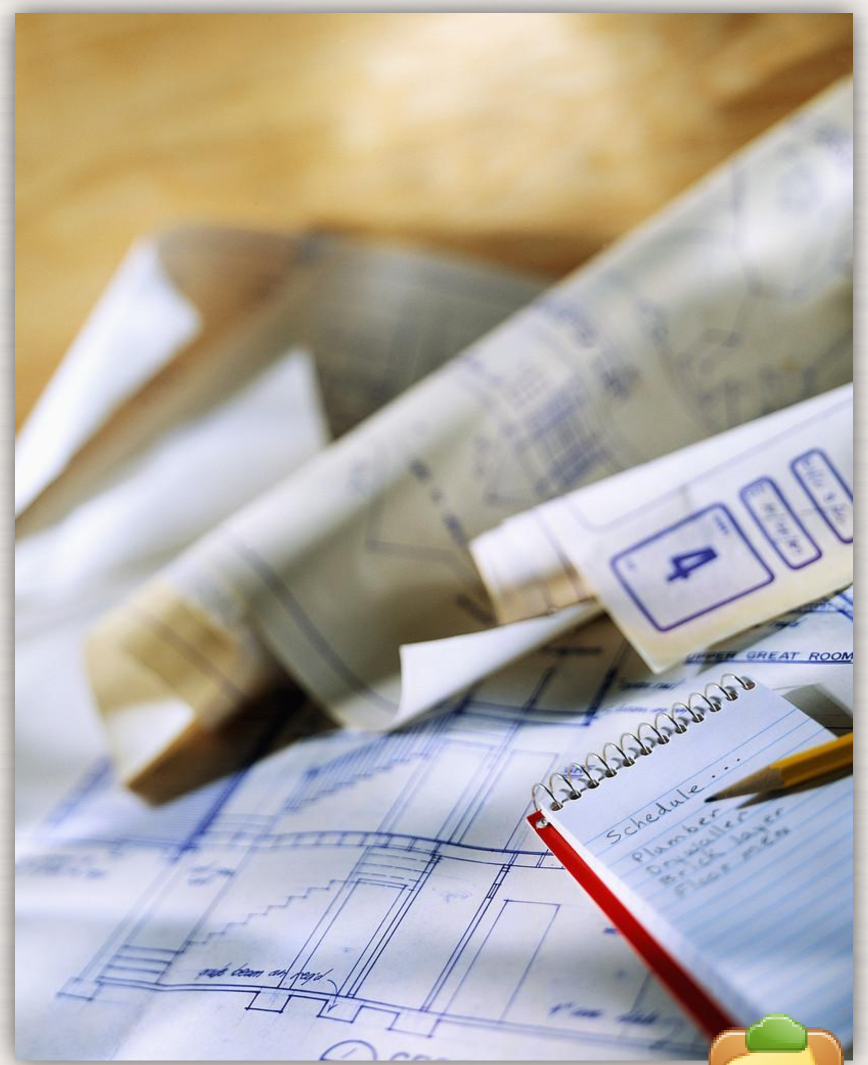
Maciej Olewiński

Cryptographic Seminar

24.04.2013r.

Agenda

- Modern applications
  - Role of networking
  - Broadcast communication
- Broadcast encryption
  - Basic information
  - Requirements
- Example cryptosystems
  - SKDC
  - Complete key graph system
  - One-way hash tree system
- Other systems
- Efficiency comparison
- Summary



Modern applications development

- Networking as the most important keyword
  - Most of modern application uses some kind of connectivity
- Term of „group of users" becoming more and more important
  - Social networks
  - Sensor networks
  - …
- Many communication principles
  - Unicast
  - Multicast
  - Broadcast
  - …
- Looking for the best solution for content distribution
  - Less unicast implementations, wider usage of broadcast

Note:
Even though broadcast and multicast are different, in this presentation they will be considered as equal from „cryptographic" point of view

Broadcast communication

- Main idea of broadcast communication
  - Sending **exactly** the same content to the group of users simultaneously
  - Without individual addressing every single user („*there is no individual user, there is a group*”)

- Example applications
  - Television (both „*traditional*” and IPTv)
  - Distance learning
  - Grid computing
  - …

Problems:
How to distribute the key only to allowed group of users (still using broadcast)?

How to make possible to add/remove every single user for the group of allowed receivers?

- Security requirements
  - High demand for confidentiality (to limit group of receivers)
  - Traditional way for ensuring confidentiality: adding an encryption

- Problem with encryption in broadcast:
  - With encryption or not, broadcast is still broadcast: the same data is being send to all receivers
  - So, if encryption is in use, all receivers must use the same key to get the same content

Broadcast encryption (1)

- Definition
  - We consider broadcast encryption problem as distributing a secret (typically: symmetric session key sk) to a privileged group G of intended receivers

- Details (1)
  - Group U: so-called *universal users set*, contains all n users (allowed and disallowed for sharing sk)
  - Group G: so-called *privileged users set*, contains users allowed for sharing sk
  - Group R: so-called *revocation users set*, contains users NOT allowed for sharing sk
  - Each user in U belongs to either G or R
  - Formally:

$$U = \{U_i\}_{i=1}^{n}$$

$$U = G \cup R$$

  - All users in U are static, which means that each user (in both G and R) can be pre-loaded with some static private information
  - All users in U may share some public information

Broadcast encryption (2)

- Details (2)
  - Encrypting application data (traffic) with session key sk enables secure communication channel between sender and privileged users in G
  - This channel is unavailable for users in R

- Key server
  - To make sure that always only privileged users from G are able to recover application data from encrypted communication, cryptographic key server is needed
  - To address problem of join and departures in G (and thus in R) this server is responsible for updating session key sk after each groups change (and distributing it to the new G members)
  - After each sk change this server broadcasts special rekey message called α which contains the new session key sk
  - This message is composed in a special way to ensure that only privileged users from G will be able to infer sk from it (using optionally also private and public information)

Requirements (1)

- Dependability
  - Current session key sk must be fully dependent of the corresponding rekey message α
  - In other words: without receiving α broadcast from the key server any user (from both G and R) should be unable to infer current session key sk

- Statelessness
  - Current session key sk should be derived only from the current rekey message α and private/public storage
  - It must be independent of any past rekey message α

- Correctness
  - Any allowed user from G should be able to get from rekey message α exactly the same session key sk that was chosen by key server

Requirements (2)

- Exclusiveness
  - Any single user from revocation set R (disallowed user) should be unable to infer current session key sk from broadcast rekey message α
  - This requirement must be met even when user from R knows current rekey message α, algorithm used for inferring sk from α and has private/public information
- Collusion resistance
  - Users from revocation set R must be unable to infer current session key sk even when they cooperate (and no matter how they cooperate)
- Efficiency
  - Communication efficiency: broadcast rekey message α should be as small as possible
  - Storage efficiency: system should require as small data storage on both receiver side and key server side as possible
  - Computational efficiency: all required calculations on both client side (inferring session key sk from rekey message α) and key server side (preparing rekey message α) should be as easy as possible

SKDC system (1)

- SKDC – *Simple Key Distribution Center*
  - The simplest broadcast encryption system

- Basics
  - Main assumption: each user i in U has its own (individual) symmetric key k_i which is known only to this user and shared with key server (preloaded private information)
  - Every time when any user join/leave group G, key server chooses new session key sk and prepares rekey message α which is then broadcasted to all users

- Rekey message α preparation
  - For each user in G key server encrypts the new session key sk using this user's secret key k_i
  - To the result of this encryption key server adds also a label indicating to which of users from G this encryption result is intended
  - The whole α message is build of such a parts for all users in the new G. Formally:

$$\alpha = \{i, E_{k_i}(sk)\}_{U_i \in G}$$

SKDC system (2)

- Inferring session key sk from rekey message α on receiver side
  - After receiving rekey message α each user in G finds the label indicating its part of α message
  - Then, it decrypts the sk using its own individual secret key k_i
  - Users in R cannot find its own part of α and thus they cannot get sk

- SKDC is simple, but secure
  - It provides dependability, statelessness, correctness, exclusiveness and collusion resistance

- Efficiency
  - Very good storage efficiency (each user stores only one individual key k_i; server stores individual keys k_i of all users, but it is also very small amount of data)
  - Very good computational efficiency (each user performs only one symmetric decryption to get sk; server performs only $|G|$ symmetric encryptions to build α)
  - Very poor communication efficiency in bigger systems (the size of α message scales linearly with the size of group G – it must contain a part for each user in G)

Complete key graph system (1)

- Main goal of the system
  - Provide excellent communication and computation efficiency
- Basics
  - Main assumption: for n users in U, there can be 2^n possible combinations of group G composition (because for each of n users in U we have only two possibilities: it can belong to G or not)
  - Thus, each user from U belongs to half of all possible groups (2^{n-1} combinations)
- Main algorithm
  - For each possible group G composition key server pre-selects distinct session key sk (so, it chooses 2^n session keys)
  - Similarly, each user from U is pre-loaded with all session keys sk which were assigned by server to those group G compositions which this user is member of (so, each users have 2^{n-1} session keys)
- Rekey message α
  - It contains only n bits describing the composition of current group G

Complete key graph system (2)

- Inferring session key sk from rekey message α on receiver side
  - After receiving rekey message α each user reads its *n*-bit value
  - If this value indicates that this user is a member of current *G* group, it just has proper sk key preloaded and chooses it using information of bits order from α
  - Otherwise, this user does not have proper sk key and cannot participate in communication

- Efficiency
  - Complete key graph system offers extremely good communication efficiency (rekey message α has a constant size of n bits where n is a size of U)
  - It also offers ideal computational efficiency – it does not require any computation on both server and client side
  - Unfortunately, it has very poor storage efficiency, as the amount of data that needs to be stored on both server and receiver side grows exponentially with the size of U. For example, for 100 users in U it is required to store 1 267 650 600 228 229 401 496 703 205 376 keys on the server

- This system is good only for very small groups of users
  - But: when the size of group is very small, communication efficiency of SKDC is not a problem

One-way hash tree system (1)

- Basics
  - This system is intended for usage in services when user can subscribe for some fixed time slots (preferable of the same duration)
  - Each time slot has assigned its own session key sk
  - Because of this, key server does not need to respond for individual user membership change – session key sk is being changed periodically
  - For storage optimization, all session keys are logically organized – they are leafs of one-way hash tree
  - Ideal for services like IPTv etc.

- Hash tree construction algorithm
  - Two functions are needed: one for encryption [$E(x)$] and one for generating one-way cryptographic hashes [$H(x)$]
  - There is a requirement, that the length of hash produced by $H(x)$ must be twice as long as the key needed by $E(x)$ (for example, if $E(x)$ is AES-128, $H(x)$ can be SHA-256)
  - The key server chooses a seed from the space of available keys for $E(x)$; it becomes a root tree node
  - By using $H(x)$, two new keys are created from the seed and becomes children of this node
  - The tree are build iteratively in the described way

One-way hash tree system (2)

- Providing session keys *sk* to receivers
  - Receiver is being pre-loaded with the session keys of time slots that it has subscribed for
  - If receiver has subscribed for „bounded” time, it is enough to provide only the root node of the sub-tree which contains required session keys in its leafs
  - With root node, receiver is able to easily calculate all needed leafs by performing hash function $H(x)$ iteratively

- Efficiency
  - Ideal communication efficiency – rekey message α is not needed (*sk* is being changed periodically at known time intervals)
  - Very good storage efficiency (server stores only the tree structure with *sk*'s of all slots as its leafs; client stores only the subtree root nodes for subscribed time slots)
  - Very good computational efficiency (only a few hash computation needed on client side)

- The only problem: this system is not intended for general use
  - Only time-slot subscription model

Other solutions; efficiency

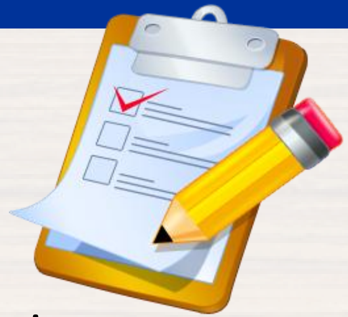
- Asymmetric solutions
  - Mostly based on RSA
  - Good storage and communication efficiency
  - Low computation efficiency

It's the use case which determines the best cryptosystem

- Efficiency comparison of presented solutions

System/Metric	Rekey message size (communication efficiency)	Storage efficiency (on client side)	Computational efficiency (on client side)
SKDC	Scales linearly with G size	One symmetric key	One symmetric decryption
Complete key graph	n-bit for n users in U	Scales exponentially with U size	No computation needed
One-way hash tree	No rekey message needed	Root nodes of keys subtree	A few hashes

Summary, conclusions



- New emerging possibilities for application of broadcast encryption
  - Rapid development of networking applications
  - Mobile devices domination on IT hardware market
  - Sensor networks
  - Group communication principle
  - Many unexplored usage areas; *trial-and error* development and *startup syndrome*

- Different available broadcast cryptosystems
  - Symmetric and asymmetric principle
  - Flexible group membership vs. time-based subscription
  - Different efficiency characteristics

- Choose of the best cryptosystem driven by use case
  - Each cryptosystem distinctly different

- Development of *all-efficient* system is still an interesting research issue

Bibliography

- **W. T. Zhu, *Towards secure and communication-efficient broadcast encryption systems***
- D. Boneh, C. Gentry, B. Waters, *Collusion resistant broadcast encryption with short ciphertexts and private keys*
- B. Briscoe, *MARKS: zero side effect multicast key management using arbitrarily revealed key sequences*
- R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, B. Pinkas, *Multicast security: a taxonomy and some efficient constructions*
- C. K. Wong, M. Gouda, S. S. Lam, *Secure group communications using key graphs*
- R. L. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public-key cryptosystem*

Thank you for your attention

Questions, comments?