Open access Journal International Journal of Emerging Trends in Science and Technology

Network Security Cryptographic Protocols and Lattice Problems

Author
Dr Daruri Venugopal
M. Tech; Ph.D; Ph.D (Post.Doc.)
Professor in Dept. of Computer Science & Engineering
Siddhartha Institute Of Technology And Science
Narapally, Ghatkesar, R.R. Dist.
Email: *Profdarurivg.edu@gmail.com*

Abstract
We present a new protocol for electronic cash which is designed to function on hardware with limited computing power. The scheme has provable security properties and low computational requirements, but it still gives a fair amount of privacy. Another feature of the system is that there is no master secret that could be used for counterfeiting money if stolen. We introduce the notion of hierarchical group signatures. This is a proper generalization of group signatures, which allows multiple group managers organized in a tree with the signers as leaves. For a signer that is a leaf of the sub tree of a group manager, the group manager learns which of its children that (perhaps indirectly) manages the signer.
Keywords: *electronic cash; security properties; counterfeiting ; computational; hierarchical*

Introduction

When the word "cryptography" is mentioned, what first comes to mind is probably sending secret messages. This is justified, as hiding information from eavesdroppers, confidentiality, is the traditional reason to use cryptography. An analogy is to send a message in a sealed envelope (or maybe in a locked safe, although it is debatable how realistic such an analogy is). Sometimes we are not primarily interested in hiding information, but rather in ensuring that information isn't modified or counterfeited, authenticity. By this we mean that the receiver can be convinced that sender is who he claims to be, and that the message has not been altered on the way. The analogy here is to sign a paper with the message on it. Since signatures are assumed to be hard to forge, a signature identifies the sender. In an environment where messages are mainly sent electronically, we need methods to achieve confidentiality and authenticity by digital means, and this is one major part of what cryptographic research is about. Since we want the system to be secure, we want it to be infeasible to compute any useful information about the plaintext from the cipher text, provided that the key k is unknown.

We even want it to be infeasible if certain side information is known, such as a subset of legal messages from which m is drawn, or encryptions of other messages under the same key. A digital signature is in one sense more secure than a physical signature on paper. When a paper with the message written on it is signed, it is hard to ensure that the message is not altered afterwards. A forger may add new text to a signed document or combine pages from two or more signed documents into a new document. A secure digital signature scheme withstands attacks of this type, since the signature is tied to the message and becomes invalid if the message is modified.

Public Key Cryptography

In the above definition, the same key is used for encryption and decryption. For a long time, this was the only known way to perform cryptography. In the middle of the 1970s, a major breakthrough was made when methods to perform asymmetric cryptography were discovered. Asymmetric systems use two keys, the public key, pk and the private key (sometimes called secret key), sk . The public key is used to encrypt, and the private key to decrypt so that D_{sk} (E_{pk} (m)) = m. The public key can be published, since it is used only for encryption, but the private key must be kept secret.

Asymmetric Encryption Schemes

Let us now compare this with symmetric cryptosystems to see what the differences may mean in practice. Assume ten people work at the same company, and that they want to be able to send encrypted messages to each other. First con-sider a symmetric cryptosystem. One solution is to have a single common key that everything is encrypted with, but there are several drawbacks with this approach. Someone who gets hold of the key (for example by bribing one of the employees) is able to read all messages sent. Also any employee can read any message, even it wasn't meant for him. If an employee quits, a new key has to be set up and distributed in a secure manner.

A second solution is to have one key between every pair of employees. Then only the intended recipient can read his messages, and if one employees sells (or accidently discloses) his keys, only the messages sent or received by that employee can be read. However, the number of keys necessary for such a system is high. Our ten employees need a total of 45 keys. Although this number may not seem very high, we must take into account that agreeing on a symmetric key is a cumbersome task.

It is not advisable to the keys electronic-ally, since they can be eavesdropped, and if a key is sent by mail, there is always the risk that someone opens the envelope and gets the key. The only safe way is to meet in person. Now consider a company with 1000 employees. Then a total of 499, 500 keys are necessary! It is obvious that symmetric cryptosystems have certain drawbacks. Now let us consider using asymmetric cryptography to solve the problem. Each of the ten employees generates a key pair consisting of a private and a public key. The public keys are published, say in the company phone book. If Alice wants to send a message to Bob, she looks up Bob in the phone book, encrypts using his public key and sends the message. Bob uses his private key to decrypt, and no-one else can read the message. If the company hires new employees, each of them generates a key pair. No keys have to be exchanged under secure conditions.

Digital Signatures

Authenticity can be achieved by asymmetric means. When a MAC is used, the same key is used for computing the MAC and verifying it. Therefore only the intended recipient can check the validity of the message. Furthermore, ability to verify implies ability to compute a MAC, making it hard to use a signature as proof in case of a dispute. Therefore, in many situations, it is desirable to have a scheme in which it is possible to verify without being able to sign. Using asymmetric techniques we can construct a scheme where the signing is performed using the private key sk and the verification with the public key pk . Now it must hold that V_{pk} (m, S_{sk} (m)) = 1. This is also what we expect from real-world signing schemes – anyone can look at a signature and check whether it has been written by the putative sender (by comparing it with other signatures written by the same person), but no-one but the sender else should be able to produce such a signature.

A digital signature is in one sense more secure than a physical signature on paper. When a paper with the message written on it is signed, it is hard to ensure that the message is not altered afterwards. A forger may add new text to a signed document or combine pages from two or more signed documents into a new document.

A secure digital signature scheme withstands attacks of this type, since the signature is tied to the message and becomes invalid if the message is modified. Two of the most important building blocks for cryptographic functions are one-way functions, i.e., functions that are easy to compute but hard to invert, trapdoor functions, i.e., functions that are one-way functions with the additional

propertythat there is a secret which makes the function easy to invert. Take, for example, multiplication. It is easy to multiply two numbers, but no method is known that factors a numbers into its prime factors in reasonable time. It should be noted that the existence of one-way and trapdoor functions is a classical open problem, and a proof of their existence would be a major breakthrough. However, there are functions that have been subject to intensive research for more than thirty years, and no evidence contradicting the hypothesis that they are trapdoor functions have been found. It is therefore reasonable to assume that they are indeed trapdoor functions. From functions that are assumed to be trapdoor functions, it is possible to build cryptographic primitives, e.g., encryption and signature schemes.

Efficient vs. Practical Protocols

Naturally we want our protocols to be as efficient as possible. However, in different contexts efficiency may have different meanings. The common definition of an efficient algorithm is that the execution time is bounded by a polynomial in the size of the input. For example, the grade school algorithm for multiplication is polynomial time, since the number of steps needed is less than $2n^2$, where n is the number of digits of each factor.

An example of an algorithm that is not polynomial is factoring by exhaustive search. To factor an n-bit number m we may need to check each number up to $\sqrt{m}$, that is, $2^{n/2}$ different numbers. Even if we assume that we can check divisibility in a single step, we still need an exponential number of steps before we are guaranteed to have a result.

It is clear that this definition of efficient algorithms does not cover everything we need from an algorithm to be usable in practice. If we design an algorithm that runs in n^{30} steps, it would still be considered efficient according to the above definition. However, the algorithm would be impossible to use in practice except for extremely small inputs.

In this thesis we focus on protocols that are not only efficient in the above meaning, but that are practical. Therefore the protocols must be specified in such detail that it is possible to analyze their running time precisely and not only show that it is bounded by some polynomial. Also, being practical is not a strict definition. In some cases, we want a protocol that can be executed on devices with little computing power such as smart-cards or mobile phones. In other cases it is enough if the protocol runs reasonably fast on a personal computer, and in still other cases the protocol will run on a server with large storage capabilities.

Security of Cryptographic Primitives and Protocols

Obviously we want the cryptographic primitives we use to be secure. However, we need to define precisely what we mean by security of a primitive.

Let us consider an encryption scheme. One definition of security is that the scheme is secure if an attacker who sees a cipher text cannot recover the plaintext. However, in some scenarios this is not enough, since the attacker may have access to additional in-formation. Maybe the attacker knows that the plaintext is either “yes” or “no”, and maybe the attacker has seen encryptions of other plaintexts. Maybe the attacker even has seen encryptions of “yes” and “no”. —

A good cryptosystem should remain secure even under these circumstances. For example, to remain secure even if the attacker knows encryptions of “yes” and “no”, the encryption must be probabilistic.

Anonymity

Designing protocols that are as secure as the primitives used is not trivial. It may very well be the case that a protocol turns out to be insecure although all components used are secure. Also in the case of protocols, the term “secure” must be properly defined. Take, for example, a scheme for electronic cash involving customers, merchants and a bank. Naturally a customer should not be able to counterfeit money, but what happens if a customer and a merchant collaborates to produce counterfeit money? Or maybe when two customers together try to create a coin that appears to be valid to the merchant but which is rejected by the bank? Obviously there are many subtle details when deciding what kind of security we want from a protocol. Therefore it is important to make a clear

definition of security and to prove that the protocol fulfills those definitions under some plausible assumptions.

Assume the cash you withdraw had your name on it. What would that mean? In most cases it wouldn't mean anything. No-one would be interested in knowing that it was you who bought that pack of chewing gum. You might feel a little bit uncomfortable if you knew that a curious trainee working in the pharmacy can keep track of what medicine you use. If the government can figure out your political viewpoint by monitoring what newspapers you purchase and what events you buy tickets to, you have reason to be really worried.

We often take anonymity for granted. If you purchase a newspaper with cash, it is not possible to trace the purchase back to you by looking at the coins you paid with. If you buy a couple of tokens for the metro, it is not possible to see if two trips were paid by tokens purchased at the same time. The simple reason neither coins nor metro tokens are traceable is that they don't have a serial number.

Payment Systems

When making purchases, the most common ways to pay for the goods is either by using cash or by using a payment card or check. Cash has the property that it is anonymous and that it is possible to verify that it is valid by just looking at it and without calling the bank. This offline property of cash is important, and very desirable. It reduces communication costs, it makes the scheme more robust since it doesn't require the bank to be available, and it is fast. The merchant can deposit the cash with his bank, use it as change, buy goods, pay salary etc. Unfortunately cash also has the not so nice property that it can be stolen.

A payment card or check, on the other hand, is not itself a proof that the customer has the money to pay. The issuer must be contacted to verify that the customer has the necessary funds, but once the transaction is completed, it cannot be stolen like cash. Since the merchant's name is part of the payment, no-one else can get credited for the transaction. Digital payment systems try to mimic these properties. Systems for digital cash try to keep the anonymity of the customer, possibly with a trusted party that can revoke the anonymity.

However, since a digital coin is just a bit-string, it can be copied and spent twice. The most common way to deal with this is to design the system so that the identity of the owner is revealed if the same coin is spent twice. Another solution is to make the system online, but then part of the motivation to use coins is lost. Systems for digital cash often require that the merchant deposits the cash with the bank after the transaction rather than reuse it.

EMV Payment Cards

Still the majority of payment cards are equipped with a magnetic stripe where the cardholder data is encoded. Although a convenient and cheap solution, it has its security problems. The magnetic stripe can be copied and modified, making it a good target for counterfeit and fraud. The transactions made with a magnetic stripe are not digitally signed, making it possible to modify the transaction data after the transaction took place.

One alternative to the magnetic stripe is smart-cards. A smart-card is a tiny computer placed on a plastic card. As with any computer, it can store and process data. It can also have some parts of its memory protected from direct access. This is a very useful property to prevent copying and modification of cards.

Since the amount of money lost on fraud by the payment networks is growing, there is an on-going program to switch to smart-cards. The switch is currently in progress, with some issuers already issuing smart-cards, and some still using the magnetic stripe. With smart-cards, the security is increased considerably. A smart-card cannot be copied or modified the same way a magnetic stripe can. It can hold secret data used only internally by the card. Smart-cards can sign transactions, thus ensuring they are sent to the payment network unmodified. Some smart-cards also contain a private key for authentication purposes. Since the private key is accessible only to the internal smart-card software, such a card cannot be duplicated.

Cryptographic Lattices

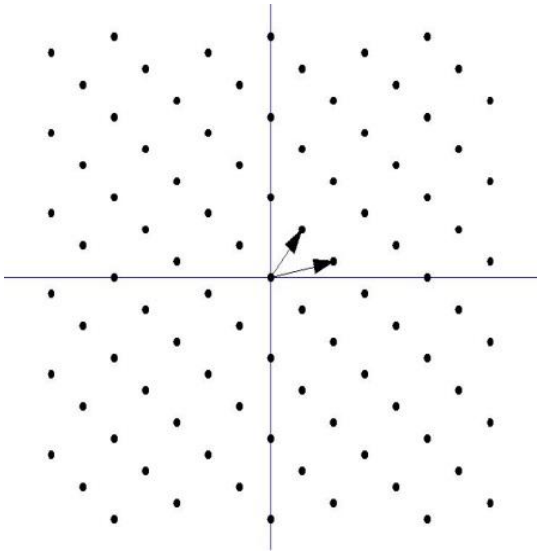

Figure 1.1: A two-dimensional lattice

In this thesis we point out a vulnerability in some EMV cards. Although the EMV standard builds on primitives in which no vulnerabilities are known, we show that certain EMV card configurations are insecure. The vulnerability would allow an attacker to use an EMV card to perform an unlimited number of offline transactions. EMV does allow for offline transactions, but there is a limit on the maximum number of consecutive offline transactions stored on the card. In Chapter 4 we show how to perform the attack, and also, where it is possible, how to configure a card to protect against the vulnerability.

Conclusions and Recommendations

We have demonstrated how EMV cards with a certain configuration can be at-tacked, and we have also pointed out how to configure a card correctly to avoid this attack. We have seen that cards based on M/Chip cannot be configured in the proposed way, and are therefore always susceptible to the attack. Here the only solution is to move to (more expensive) DDA cards.One possibility to solve the problem is the change the EMV specification so that a terminal always goes online when a non-DDA EMV card is used. Although the consequence is that issuers using low-cost card cannot benefit from the advantages of offline transaction, from a security perspective this approach would be the most efficient.

References

1. M. Ajtai. Generating hard instances of lattice problems. In 28th ACM Symposium on the Theory of Computing (STOC), pages 99–108. ACM Press, 1996.
2. M. Ajtai. The shortest vector problem in ℓ_2 is NP-hard for randomized reductions. In 30th ACM Symposium on the Theory of Computing (STOC), pages 10–19. ACM Press, 1998.
3. G. Ateniese, J. Camenisch, M. Joye, and G. Tsudik. A practical and provably secure coalition-resistant group signature scheme. In Advances in Cryptology – CRYPTO 2000, volume 1880 of Lecture Notes in Computer Science, pages255–270. Springer Verlag, 2000.
4. G. Ateniese and G. Tsudik. Some open issues and directions in group signatures. In Financial Cryptography ’99, volume 1648 of Lecture Notes in Computer Science, pages 196–211. Springer Verlag, 1999.
5. L. Babai. Trading group theory for randomness. In 17th ACM Symposium onthe Theory of Computing (STOC), pages 421–429. ACM Press, 1985.

Print References

6. EMVCo. EMV2000 Integrated Circuit Card Specifications for Payment Systems, December 2000. Available fromhttp:// www.emvco.com (September203).
7. U. Feige, D. Lapidot, and A. Shamir. Multiple non-interactive zero-knowledgeproofs under general assumptions. SIAM Journal of comp. 29(1):1–28,1999.

Author Profile

Dr. Daruri Venugopal Received the Bachelors and M.Sc. degrees in Mathematics from Osmania

University in 1995 and 1997, respectively. He done his M.Phil Mathematics from Algappa University in the year 2003. He Done his Doctorate from NITK, Surathkal in the year 2006 in Computer Science Engineering. He done his M.Tech Computer Science Engineering from JRN Deemed University in 2007. He has over 110 Research Paper to his Credit. He is Editorial reviewer for four Reputed International Journals in Mathematics & Computer Science Areas. He is a Recognized Ph.D Supervisor in the Areas of Mathematics and also in Computer Science and Network Engineering. Presently working as Professor in Siddhartha Institute of Technology and Sciences, Hyderabad.