A New Family of Authentication Protocols

Ross Anderson[1], Francesco Bergadano[2], Bruno Crispo[2], Jong-Hyeon Lee[1], Charalampos Manifavas[1] and Roger Needham[3]

[1] Cambridge University Computer Laboratory, England
[2] Dipartimento di Informatica, Università di Torino, Italy
[3] Microsoft Research, Cambridge, England

Abstract. We present a related family of authentication and digital signature protocols based on symmetric cryptographic primitives which perform substantially better than previous constructions. Previously, one-time digital signatures based on hash functions involved hundreds of hash function computations for each signature; we show that given online access to a timestamping service, we can sign messages using only two computations of a hash function. Previously, techniques to sign infinite streams involved one such one-time signature for each message block; we show that in many realistic scenarios a small number of hash function computations is sufficient. Previously, the Diffie Hellman protocol enabled two principals to create a confidentiality key from scratch: we provide an equivalent protocol for integrity, which enables two people who do not share a secret to set up a securely serialised channel into which attackers cannot subsequently intrude. In addition to being of potential use in real applications, our constructions also raise interesting questions about the definition of a digital signature, and the relationship between integrity and authenticity.

Keywords: authentication, non-repudiation, hashing, timestamping

1 Introduction

Most existing cryptographic protocols that provide non-repudiation, whether of origin or receipt, are based on digital signature algorithms such as RSA and DSA. However convenient this may be in some applications, is not necessary: nonrepudiation services have been provided without signatures, and an example is the SWIFT system for international banking transactions that was fielded in the mid 1970's as a replacement for older and less secure telegraphic transfer systems. In SWIFT, pairs of corresponding banks shared MAC keys that were exchanged manually; the messages passed from one bank to another over a private network with multiple independently administered logging facilities (the latest version of SWIFT has digital signatures but since they are apparently applied to the MAC, the logging service is still needed for non-repudiation). However third party logging facilities are expensive and in many applications it is desired that principals should have the means to generate and store evidence that they might need to press their claims in subsequent disputes.

An early system that did not rely on third party logging was designed by TRW for the NSA in the 1970's to authenticate messages from sensors placed in missile silos to monitor the SALT 2 treaty [1]. It used concatenated encryption: a message would first be encrypted using a Russian device and then with an American one. The keys were made available to the other side (and to third party monitors such as the United Nations) after the messages had been logged by all interested parties. This technique was published in 1983 in the context of the test ban treaty; at that time, it was preferred over RSA because neither of the two superpowers would trust a device built by the other, and asymmetric mechanisms were felt to offer little additional benefit given this constraint [2].

The first published approach to the provision of nonrepudiation without using asymmetric encryption is due to Lamport, who generates signatures by opening commitments that have been made using a one-way hash function [3]. The basic idea here is that the signer chooses two random numbers (representing 0 and 1) for each bit of the message, and publishes their images under a one-way hash function. To sign a message, he reveals the preimages corresponding to the actual 0's and 1's. Despite refinement by Merkle [4, **?**], by Even, Goldreich and Micali [6] and by Bleichenbacher and Maurer [7], this technique still requires a lot of hash computations.

In this paper, we show how to construct digital signatures that require only a small number of hash function computations each. This enormous improvement is brought about by making signature interactive; users may either interact with each other or with a time-stamping service. In many applications, interaction with a counterparty or a trusted third party service is a requirement in order to verify the availability of funds, the uniqueness of negotiable instruments or the absence of a key on a certificate revocation list. In this case, it may be possible to dispense with signatures based on number theory.

Possible benefits include the elimination of some patent royalty and export control problems and a measure of insurance against any success that quantum computers have in breaking systems based on number theory. In addition, as there are no long term secrets in our protocols, they might help to overcome intelligence agency concerns that signature keys can easily be abused for encryption. Finally, our constructions raise a number of interesting theoretical points.

2 The Basic Idea

The underlying idea came to us on the 4th November 1996 while discussing how a modern day Guy Fawkes, about to blow up the Houses of Parliament, could arrange publicity for his cause — but without getting caught[1].

[1] for the benefit of readers unfamiliar with British history, Fawkes conspired to blow up King James the first and the Houses of Parliament in 1605; this was an attempt

The naïve approach might be to telephone the newsroom of the 'Times' and say *"I represent the free Jacobin army and we are going to carry out a liberation action tomorrow. Once we have done it, I will call using the code word 'Darnley' to state our demands"*.

2.1 Using hash functions

This is not a particularly secure way of doing things. The message will be passed on to the police, who might remember that the state opening of Parliament is imminent and double the guard. So it would be more prudent to send a hash of the message. Provided that the hash function is truly one-way (technically, pseudorandom), this will not leak information. If Fawkes is now successful, he can reveal the message and find himself possessed of a very credible codeword.

We understand that organisations such as the IRA do in fact share codewords with newspapers and use these to claim credit for their crimes. However, such a protocol is open to abuse by newspaper staff (as well as by other people with access, such as phone company employees and policemen authorised to tap telephone lines). So our next logic step in improving the protocol is to replace the codeword with its hash.

Our protocol is now

- Select a random codeword X
- Form its hash $Y = h(X)$
- Construct a message $M =$ *"We are the free Jacobin army and we are going to blow up the Houses of Parliament tomorrow. The codeword by which we will authenticate ourselves afterwards will be the preimage of* Y*"*
- Compute $Z = h(M)$ and publish it anonymously
- Blow up the Houses of Parliament
- Reveal M

This might appear to be only a slightly more technological version of the protocols already used by various liberation groups and criminals. It still suffers from the serious problem that, in the face of a capable motivated opponent, the password is only one-time; once it has been revealed, and is known to the newspapers and the police, any journalist or policeman could in theory masquerade as the rebel leader. Indeed, if Guy Fawkes tries to state his demand by sending the 'Times' the codeword X with a political demand P ('votes for Catholics!'), the police can intercept the message and replace P with the demand P' ('a million guineas for Fawkes!'), thus discrediting Fawkes and his organisation.

to end the persecution of Roman Catholics. Fawkes was caught as a result of a comsec failure (a coded letter from one of the conspirators was intercepted and deciphered). After his public execution, Parliament ordained the 5th of November as a day of thanksgiving for their narrow escape, and it is still celebrated by bonfires and fireworks displays.

2.2 The Guy Fawkes Protocol

Our critical innovation is to introduce a chaining mechanism that lets us bind codewords to messages in a way that provides not just authentication but also nonrepudiation. It also allows the secret codeword to be refreshed, so that the system can be used an arbitrary number of times.

The basic idea is that, at each round of the protocol, we firstly commit to a string consisting of (codeword, message, [hash of next codeword]) by publishing a hash of it. This commitment binds the message to the codeword and its successor. We then reveal the value of this string, proving our knowledge of the codeword and thus authenticating ourselves.

Formally, we define the protocol by induction. Suppose that we have published Z_i followed by the message M_i containing $h(X_i)$, where our secret codeword is currently X_i. We wish to authenticate the message M_{i+1}. We follow the following protocol:

– Select a random codeword X_{i+1}
– Form its hash $h(X_{i+1})$
– Compute $Z_{i+1} = h(M_{i+1}, h(X_{i+1}), X_i)$ and publish it
– Reveal $M_{i+1}, h(X_{i+1})$ and X_i

The first codeword needs to be bootstrapped by some external mechanism; in most applications, this would be a conventional digital signature or an out-of-band authentication, perhaps using a conventional CA. We will give some examples below.

3 Discussion

Hash chains were introduced by Lamport [3] and have been used in one-time password applications such as the S/Key one-time password system [8], as well as several electronic payment protocols [9–11]. There, as here, the effect is to establish secure association at low computational cost.

In S/Key, a user has a series of one-time passwords, each of which is the preimage of its predecessor; the goal is to show that an authorised user is present and protect against passive attacks (though not against active attacks such as session stealing). This chain of events is rooted in a single manual authentication event in which the last element of the hash chain is set as the first password in the system.

In the payment protocols, the goal is to associate a number of electronic coins with a single digital signature that authenticates them all, and thus enable a series of small payments to be made by a customer to a single merchant (such as a phone company) at the cost of a single digital signature or online authentication operation.

In the Guy Fawkes protocol, the objective is to associate a single act of authentication with a stream of future statements rather than a stream of future events. Functionally, the difference is that while the format of all the digital coins is known at the time they are signed, the future statements that we wish to authenticate may not be. So it would not be sufficient to simply use a hash chain (as in S/Key) as a set of one-time passwords for authenticating political statements. As in section 2.1 above, anyone who was tapping the line when the statement and password were sent to the newsroom could alter the statement; and staff in the newsroom could also substitute messages at will.

In other words, the broadcast commitment step has the criticial effect of providing nonrepudiation, and gives the Guy Fawkes protocol the same effect as a digital signature. Were the Jacobins permitted to use asymmetric cryptography, then their first message could just as well have read *"We are going to blow up the Houses of Parliament on the 5th November. Future demands will be digitally signed and the public verification key is W."* This could have been encrypted and published, with the key made known after the event.

So we might ask whether there is anything to signature other than secure association. After all, in the conventional model, a digital signature sets up a secure association between something that has been signed at an arbitrary time, and an authentication instance which may have involved showing a passport to a certification authority. Is the Guy Fawkes protocol any different?

4 It may be secure, but it is a signature?

At this point, some might argue that although the Guy Fawkes protocol gives the same effect as a digital signature, it is not actually a signature. However, our protocol satisfies most of the definitions of 'digital signature' offered in the literature to date. We will go through them in turn.

Diffie and Hellman introduced the concept of digital signature in their seminal 'New Directions' paper: '*it must be easy for anyone to recognise the signature as authentic, but impossible for anyone other than the legitimate signer to produce it*' [12]. At the time when this paper was written, the only known way of doing this was using Lamport's one-time signature. The Guy Fawkes protocol improves on Lamport and so, not surprisingly, satisfies this definition; it also satisfies a later definition by Diffie as '*a way of demonstrating to other people that (a message) had come from a particular person*' [13].

Fiat and Shamir refine and extend the definition given by Diffie and Hellman. Authentication is when A can prove to B that she's A, but no-one else can prove to B that he's A; identification is when A can prove to B that she's A, but B cannot prove to anyone else that he's A; and signature is when A can prove to B that she's A, but B can't even prove to himself that he's A [14]. Guy Fawkes satisfies this definition too.

Goldwasser, Micali and Rivest give a more involved description that explicitly mentions a number of algorithms and their properties: a key generation algorithm, a signature algorithm, and a verification algorithm. The signature algorithm produces a signature on being input the message, the key and possibly other information (such as a random input); however, in their definition it produces only a single output [15].

This model therefore excludes the Guy Fawkes protocol. But it also excludes the large class of arbitrated signatures that were already well known and in use by that time (see, for example, [16]) as well as most of the special purpose signature constructions that require interaction, such as undeniable signatures, designated confirmer signatures and oblivious signatures [17].

Naor and Yung refined the approach of Goldwasser, Micali and Rivest, by cutting the complexity theoretic requirement of the construction [18]; it was finally reduced by Rompel to the existence of one-way functions (which is minimal) [19]. However, like Goldwasser, Micali and Rivest, their definitions also fail to deal with signatures that use interaction.

Pfitzmann provides the most thorough study of disparate signature schemes in her thesis [20]. She concludes that the general definition of signature is a process with a number of access points — typically for the signer, the recipient and the court. Time is a necessary component, although logical time (in the sense of a 'global notion of numbered rounds') is sufficient (*op. cit.*, p 54). Special access points can be added for risk bearers such as certification and revocation authorities. This definition clearly admits the Guy Fawkes protocol.

So if it is claimed that the Guy Fawkes protocol is not really a signature, then the onus would be on the objector to show how to deal with the many other kinds of signature that use interaction, as well as the importance of context — the framework of certification and revocation services, legal conventions and so on — to the utility of digital signatures. In most applications, the value of signatures ultimately depends on convention (such as a digital signature law, or a contract between members of an EDI system) and the validation of even conventional digital signatures involves reference to an online or at least near-real-time certificate revocation list.

In passing, we observe that the signatures produced by Guy Fawkes are actually stronger than RSA in the sense that they can be fail-stop at no extra cost: just choose the secrets X_i uniformly at random as bitstrings significantly longer than the hash function's output. That way, an attacker who finds a preimage of a commitment or hashed codeword will with high probability have found a different one from that known to the genuine signer, who will thus be able to exhibit a collision for the hash function.

5 Signing Bidirectional Digital Streams

Hash-based signatures have been condemned as "time-consuming, costly and wasteful" ([16]). Guy Fawkes is much less so than previous schemes; and there are applications for which it might be practical.

Firstly, let us consider the most convincing proposal for a practical application of hash-based signatures. This is the method of Gennaro and Rohatgi for signing digital steams [21]. When signing a stream whose content is not known in advance to the signer (such as a television programme), they divide the stream into blocks; each block contains a one-time public key using the Lamport scheme, and is signed with the one-time private key whose public key was sent in the previous block. The first block is signed using a conventional mechanism such as RSA. In this way, a single conventional signature can be leveraged to sign a whole stream of data 'on-the-fly'. The authentication thus provided is fast, in that no use is made of asymmetric cryptography once the session is established; but it is bulky, as both a one-time public key and a one-time signature must be added to each block.

The Guy Fawkes mechanism can be adapted readily to this application and can greatly reduce the amount of computation required; it can cope particularly well with bidirectional streams, such as in videoconferencing, although it also works well in applications where a stream is sent to a recipient who merely sends a series of acknowledgements.

Here, our protection goals are that if any bit in the two streams is changed, both communicating parties will detect the problem; and that the authentication mechanisms are as fast as in Gennaro and Rohatgi's scheme without the message extension (in fact Guy Fawkes is faster). Finally it must provide non-repudiation as well as simple authentication; a third party observing the stream exchange can ascertain the information source and integrity, as opposed to symmetric MACs where the use of shared secrets makes mutual recrimination possible.

In this protocol, Alice and Bob will exchange message streams consisting of sequential blocks which we will call $A_0, A_1, A_2, \ldots$ and $B_0, B_1, B_2, \ldots$ respectively; each block will be accompanies by authentication information to be described. B_i is sent after A_i but before A_{i+1}.

In addition, Alice will choose a series of passwords X_0, X_1, X_2, ... ; she will commit to X_i in message A_{i-1} and reveal it in message A_{i+1}. This commitment is called a_i and has the form

$$a_i = h(A_{i+1}, h(X_{i+1}), X_i)$$

Similarly, Bob's commitments take the form $b_i = h(B_{i+1}, h(Y_{i+1}), Y_i)$. It should be noted that Alice needs a buffer size equal to two blocks; to send message A_i she needs to know A_{i+1} in order to compute the hash value a_i. This will not normally be a problem where, for example, each block is a frame of video.

The first steps of the protocol, which use conventional signatures to bootstrap the process, run as follows:

$A \longrightarrow B : A_0, a_0, h(X_0)$,$\mathrm{sign}_A(A_0, h(X_0))$
$B \longrightarrow A : B_0, b_0, h(Y_0)$,$\mathrm{sign}_B(B_0, h(X_0))$
$A \longrightarrow B : h(b_0, X_0)$
$B \longrightarrow A : h(a_0, Y_0)$

The authentication of each subsequent block now takes the following form:

$A \longrightarrow B : A_1, a_1, h(X_1), X_0$
$B \longrightarrow A : B_1, b_1, h(Y_1), Y_0$
$A \longrightarrow B : h(b_1, X_1)$
$B \longrightarrow A : h(a_1, Y_1)$
...

Thus in this step, Alice has committed to the password X_2 (since $a_1 = h(A_2, h(X_2), X_1)$ and revealed the password X_0; this revelation authenticates A_1, while the commitment also refreshes the passwords.

The security of this scheme follows inductively. Assuming faithful execution up to step n, and an attacker who tries to masquerade as Bob to Alice, having seen and intercepted the string $B_n, b_n, h(Y_n), Y_{n-1}$. He cannot change B_n as b_{n-1} contains a commitment to it; he cannot change b_n as it contains as an input Y_n, which he doesn't know but which was committed in b_{n-1}; $h(Y_1)$ was similarly committed in b_{n-1}; and if he forwards anything other than the correct value of Y_{n-1} then this will fail to verify against b_n and b_{n-1}. Similarly, he cannot in the next message forward anything other than the correct value of $h(a_n, Y_n)$ as he does not know the value of Y_n yet cannot change it, since it was committed at the previous step in n_{n-1}.

As a corollary, we obtain a protocol for authenticating a single digital stream: Alice sends the stream to Bob, and Bob simply sends an ack with a serial number as the text B_n.

There is one final subtlety. If all the passwords are eventually made known, then false content can be cut and pasted at will into a record of the exchange. In the basic protocol, we thus have something weaker that a signature, but stronger than symmetric authentication. This is an interesting fact in itself; an obvious direct application is witnessed communication, where (for example) a videoconference is also viewed by a third party who may be called on to testify about some aspects of it later. Another is in communication systems with third party logging, such as the SWIFT system mentioned above.

However, our scheme can be converted quite simply into one with off-line nonrepudiation. The trick is a convention that each principal keeps secret their last password and reveals it to a judge in the event of a dispute. Alternatively, each principal could have a notary sign a hash of his last password together with a transcript of the session.

6 Other Practical Applications

We now consider a number of other applications of our technique.

6.1 An Integrity Equivalent of Diffie-Hellman

Our protocol allows us to link a number of incidents securely, and so we ask whether it has any particularly interesting uses for the identification of principals in computer networks. After all, a principal is in some sense just the linkage of a series of incidents.

In the real world, it is often only necessary to remember a certain distance back in such a chain. We may be quite unable to remember what incident first convinced us of the identity of our mother, or of many of our other relatives and friends; but the absence of a definitive initial authentication instance is considered to be irrelevant in such circumstances. Similar considerations may apply to electronic personae. Many people nowadays have built up relationships and even scientific collaborations over the net with other people whom they only later meet in person.

The above protocol for bidirectional authentication shows how we can interlock hash-based authentication by two different individuals at the same time. One novel implication of this is that two principals who do not originally share any secret can protect the serialisation of the traffic between. Once this integrity channel is established, it will guarantee both the content and the correct serialisation of all future messages.

This channel does for integrity what the Diffie Hellman protocol does for secrecy. This may seem counterintuitive, and it certainly challenges the common understanding that the '*man-in-the-middle attack can defeat any protocol not involving a secret*' [17]. What is actually happening of course is that a middleperson attacking the integrity channel has to participate in it from the start; she cannot join in later, or the views that the two participants of the transaction history will differ in nontrivial ways.

At the systems level, this is because we have set up a channel with integrity but no authenticity, in the sense that we do not know who we are speaking to. So Alice, who wanted to speak to Bob, might in fact be speaking to Charlie. However in this case Charlie will have to participate actively in the conversation between them for the rest of time if he wishes to escape detection; he will not be able to drop in and out of their traffic at will.

There are applications in which conventional authentication may not be possible and yet the limitation on active attacks that this technique provides might be valuable. An example is communication between dissidents in an oppressive country that compels the escrow of even signing keys. In such conditions, trust is likely to be built up slowly over a long series of messages, and users may well wish to be sure that a channel that they are starting to trust is not taken over by authority.

A curious feature of multiparty Guy Fawkes, however, is that when one principal introduces two others with whom he has established sessions, he cannot ever persuade either of them that the other actually exists. Alice, on being introduced to Bob by Charlie, could just as easily be introduced to another persona of the principal behind Charlie. This appears to be a feature of electronic communications in general; it is merely brought out when we start to consider protocols for establishing trust that do not rely on some bootstrapping event in the physical world. (The Rivest-Shamir interlock protocol is the only one we know of that can achieve a similar effect [24]; but previous comment on it has focussed on the understandable difficulty of using it for authentication [25].)

6.2 Tamper-evident audit trails

It is a well known problem that an intruder can often acquire root status by using well known operating system weaknesses, and then alter the audit and log information to remove the evidence of the intrusion. In order to prevent this, some Unix systems require that operations on log and audit data other than reads and appends be carried out from the system console. Others do not, and it could be of value to arrange alternative tamper-evidence mechanisms.

A first idea might be to simply sign and timestamp the audit trail at regular intervals, but this is not sufficient as a root intruder will be able to obtain the private signing key and retrospectively forge audit records. In addition, the intervals would have to be small (of the order of a second, or even less) and the computation of RSA or DSA signatures at this frequency could impose a noticeable system overhead.

In this application, the Guy Fawkes protocol appears well suited because of the low computational overhead (two hash function computations per signature) and the fact that all secrets are transient; this second's secret codeword is no use in forging a signature of a second ago.

The envisaged architecture here is that each server or other sensitive machine on a LAN would authenticate its log and audit data once per second (or even more frequently) with a local timestamping service, that would run on a machine stripped of vulnerabilities such as sendmail. This could in its turn interact at some suitable frequency with an external machine such as a corporate time stamping service, which in turn could interact with a commercial service. The protocols for this are currently under development.

6.3 Secure access to timestamping services

Third party timestamping services have much wider uses than simply providing trust backup for audit data; they are used to provide evidence of priority for all kinds of intellectual property, financial records and other business documents. An example design of such as service is found in that of Haber and Stornetta [23]. There the messages to be stamped are hashed in a tree, with a hash of all input messages being made available once a second over the web and once a week through a newspaper advertisement. A signer can incorporate the relevant part of this hash tree with the disclosure of his message in the same way that he would incorporate a collection of certificates.

This immediately raises the question of how the user can trust that the timestamping service that appears to have incorporated her message into its tree is a genuine one, and not a simulacrum created by an attacker who has taken over her network connection. A conventional approach would be for the timestamping service to affix a digital signature to the timestamps it returns. However, this brings extra complexity into the trust loop, with possible attendant costs of licensing a digital signature technology. There are also performance issues in signature generation when providing a service sized to generate a thousand timestamps a second, as Haber and Stornetta's system is; this can be provided on a workstation which will however only generate 50 RSA signatures per second.

The simple solution is to use the Guy Fawkes protocol as a means of authenticating the timestamping service to the user. We are currently working on an implementation that will work with this timestamping service.

6.4 Other applications

Other specific applications in which the Guy Fawkes protocol might offer advantages over other integrity and nonrepudiation mechanisms include the updating of root keys used by software vendors and CAs; telegambling; digital elections; membership of clubs with optional anonymity; and software metering mechanisms in which a vendor sends 'keep-alive' messages to the systems of those subscribers who keep on paying their licence fees. The value that Guy Fawkes can provide here lies in the absence of a single short long term secret that a pirate could broadcast.

Another family of applications is in general authentication and non-repudiation protocols where for cost reasons it is desired to use low-power processors, such as cheap smartcards or microcontrollers. In general, where we have an interactive application in which some combination of anonymity with either serialisation or temporary nonrepudiation is required, a protocol based on Guy Fawkes may be the tool for the job.

Finally, given the politics of cryptography, it may be worth remarking that all secrets in the Guy Fawkes protocol become known; there are no long-term user secrets that can be used as decryption keys and thus less motive to attempt to escrow signing keys, with the consequential loss of evidential reliability. It can of course be used to detect middleperson attacks on Diffie Hellman key exchange, and thus to set up confidential channels indirectly. However it is unclear that any nonrepudiation — or even authentication — can be achieved if preventing authentic Diffie-Hellman is a national policy imperative; even a simple password is enough to prevent middleperson attacks [22].

7 Conclusion

We have shown that it is possible to provide a non-repudiation service without public key mechanisms, tamper resistance or third party logging. We do not even require any principal to possess a long-term secret.

This involves a new protection primitive which in its simplest form behaves extremely like a digital signature and may be obtained at a negligible computational cost, provided that a timestamp service exists. We have also shown a bidirectional primitive that may be used to authenticate digital streams at much less cost than the previous best protocol. This led us to an 'integrity equivalent' of Diffie Hellman: two users can under quite reasonable assumptions establish a channel whose traffic is protected against modification, without either of them possessing a secret at the start of the protocol or concealing any secrets from authority.

Quite apart from possible applications, these constructions raise a number of interesting questions, such as: what exactly is a digital signature? what is the necessary role of communication in a public key infrastructure? and what tradeoffs are there between computation, communication and the maintenance of state?

References

1. "The History of Subliminal Channels", GJ Simmons, in *Proceedings of the First International Workshop on Information Hiding* (Springer LNCS v 1174) pp 237–256
2. "Verification of Treaty Compliance — Revisited", GJ Simmons, in *Proceedings of the IEEE Symposium on Security and Privacy* (IEEE, 1983) pp 61–66
3. "Constructing digital signatures from a one-way function", L Lamport, *SRI TR CSL 98* (1979)
4. "A Digital Signature Based on a Conventional Encryption Function" RC Merkle, in *Advances in Cryptology — Crypto 87* (Springer LNCS v 293) pp 369–378
5. "A Certified Digital Signature", RC Merkle, in *Advances in Cryptology — Crypto 89* (Springer LNCS v 435) pp 218–238
6. "On-line / off-line digital signatures", S Even, O Goldreich, S Micali, in *Advances in Cryptology — Crypto 89* (Springer LNCS v 435) pp 263–275
7. "Directed Acyclic Graphs, One-way Functions and Digital Signatures", D Bleichenbacher, UM Maurer, *Advances in Cryptology — Crypto 94* (Springer LNCS v 839) pp 75–82
8. "The S/KEY One-Time Password System", N Haller, in *Proceedings of the ISOC Symposium on Network and Distributed System Security* (February 1994, San Diego, CA) pp 151 - 157; see also RFCs 1704, 1760 and 1938
9. "NetCard — A Practical Electronic Cash System", R Anderson, C Manifavas, C Sutherland, in*Proceedings of the Fourth Cambridge Security Protocols Workshop* (Springer LNCS v 1189) pp 49–57
10. "PayWord and MicroMint: Two Simple Micropayment Schemes", RL Rivest, A Shamir, in *Proceedings of the Fourth Cambridge Security Protocols Workshop* (Springer LNCS v 1189) 69–87

11. "Electronic Payments of Small Amounts", TP Pedersen, in *Proceedings of the Fourth Cambridge Security Protocols Workshop* (Springer LNCS v 1189) 59–68
12. "New Directions in Cryptography", W Diffie, ME Hellman, in *IEEE Transactions on Information Theory* v IT-22 no 6 (November 1976) pp 644–654
13. "The First Ten Years of Public-Key Cryptography", W Diffie, in *Proceedings of the IEEE* v 76 no 5 (May 88) pp 560–577
14. "How To Prove Yourself: Practical Solutions to Identification and Signature problems", A Fiat, A Shamir, in *Advances in Cryptology — CRYPTO 86*, Springer LNCS v 263 pp 186–194
15. "A Digital Signature Scheme Secure Against Adaptive Chosen Message Attacks", S Goldwasser, S Micali, RL Rivest, in *SIAM Journal of Computing* v 17 no 2 (April 1988) pp 281–308
16. "Digital Signatures with Blindfold Arbitrators who Cannot Form Alliances", SG Akl, in *Proceedings of the 1983 IEEE Computer Society Symposium on Security and Privacy*, pp 129–135
17. *'Applied Cryptography'*, B Schneier, Wiley 96
18. "Universal One-Way Hash Functions and Their Cryptographic Application", M Naor, M Yung, in *Proceedings of the 21st Annual ACM Symposium on the Theory of Computing* (1989) pp 33–43
19. "One-Way Functions are Necessary and Sufficient for Digital Signatures", J Rompel, in *Proceedings of the 22nd Annual ACM Symposium on the Theory of Computing* (1990) pp 387–394
20. '*Digital Signature Schemes — General Framework and Fail-Stop Signatures*', B Pfitzmann, Springer LNCS v 1100
21. "How to Sign Digital Streams", R Gennaro, P Rohatgi, in *Advances in Cryptology — CRYPTO 97*, Springer LNCS v 1294 pp 180–197
22. "On fortifying key negotiation schemes with poorly chosen passwords", RJ Anderson, TMA Lomas, in *Electronics letters* v 30 no 12 (23rd July 1994) pp 1040–1041
23. "How to Time-Stamp a Digital Document", S Haber, WS Stornetta, in *Journal of Cryptology* v 3 no 2 (1991) pp 99–112
24. "How to Expose an Eavesdropper", RL Rivest, A Shamir, in *Communications of the ACM* v 27 no 4 (Apr 84) pp 393–395
25. "An Attack on the Interlock protocol When Used for Authentication", SM Bellovin, M Merritt, *IEEE Transactions on Information Theory* v 40 no 1 (Jan 94) pp 273–275
26. "Secure Agreement Protocols: Reliable and Atomic Group Multicast in Rampart', MK Reiter, in *Proceedings of the 1994 ACM Conference on Computer and Communications Security* pp 68–80
27. "Maintaining Security in the Presence of Transient Faults", R Canetti, A Herzberg, in *Advances in Cryptology — CRYPTO 94*, Springer LNCS v 839 pp 425–438
28. "Network Randomization Protocol: A Proactive Pseudo-Random Generator", CS Chow, A Herzberg, in *Usenix Security 95* pp 55–63
29. "The Omega Key Management Service", MK Reiter, MK Franklin, JB Lacy, RA Wright, in *Proceedings of the 1996 ACM Conference on Computer and Communications Security* pp 38–47