On Bluetooth Repairing: Key Agreement based on Symmetric-Key Cryptography

Serge Vaudenay

EPFL
CH-1015 Lausanne, Switzerland
http://lasecwww.epfl.ch

Abstract. Despite many good (secure) key agreement protocols based on public-key cryptography exist, secure associations between two wireless devices are often established using symmetric-key cryptography for cost reasons. The consequence is that common daily used security protocols such as Bluetooth pairing are insecure in the sense that an adversary can easily extract the main private key from the protocol communications. Nevertheless, we show that a feature in the Bluetooth standard provides a pragmatic and costless protocol that can eventually repair privateless associations, thanks to mobility. This proves (in the random oracle model) the pragmatic security of the Bluetooth pairing protocol when repairing is used.

1 Setting up Secure Communications

Digital communications are often secured by means of symmetric encryption and message authentication codes. This provided high throughput and security. However, setting up this channel requires agreeing on a private key with large entropy. Private key agreement between remote peers through insecure channel is a big challenge. A first (impractical) solution was proposed in 1975 by Merkle [19]. A solution was proposed by Diffie and Hellman in 1976 [12]. It works, provided that the two peers can communicate over an authenticated channel which protects the integrity of messages and that a standard computational problem (namely, the Diffie-Hellman problem) is hard.

To authenticate messages of the Diffie-Hellman protocol is still expensive since those messages are pretty long (typically, a thousand bits, each) and that authentication is often manually done by human beings. Folklore solutions consist of shrinking this amount of information by means of a collision-resistant hash function and of authenticating only the *digest* of the protocol transcript. The amount of information to authenticate typically reduces to 160 bits. However, collision-resistant hash functions are threatened species these days due to collapses of MD5, RIPEMD, SHA, SHA-1, etc. [9,23,24,25,26]. Furthermore, 160 bits is still pretty large for human beings to authenticate. Another solution using shorter messages have been proposed by Pasini and Vaudenay [20] using

a hash function which resists second preimage attacks (like MD5 [21]; namely: collision resistance is no longer required) and a commitment scheme. Other solutions such as MANA protocols [13,14] have been proposed. They can reduce the amount of information to be authenticated down to 20 bits, but they work assuming a stronger hypothesis on the authenticated channel, namely that the authentication occurs without any latency for the delivery. Some protocols based on the Diffie-Hellman one were proposed [11,15] with an incomplete security analysis. A provably secure solution was finally proposed by Vaudenay [22]. This protocol can work with only 20 bits to authenticate and is based on a commitment scheme. Those authentication protocols *can* be pretty cheap (namely: without public-key cryptography) and provably secure (at least in the random oracle model). So, the remaining overwhelming cost is still the Diffie-Hellman protocol. Since key agreement is the foundation to public-key cryptography, it seems that setting up secure communications with an authenticated channel only cannot be solved at a lower expense than regular public-key algorithms.

The Bluetooth standard starts from a slightly different assumption, namely that there is a private channel between the two devices involving the human user. Of course, this channel should be used to transmit as few bits as possible. This would, in principle, be possible by using password-based authenticated key agreement. A first protocol family was proposed (without security proof) in 1992 by Bellovin and Merritt [8]. SRP [27,28] is another famous protocol, available as the RFC 2945, proposed in 1998 by Wu. The security analysis followed a long research program initiated by Bellare and Rogaway [5,6]. Specific instances of the Bellovin-Merritt protocols with security based on the random oracle model were provided in [3,4,7,10,18] starting in 2000. Finally, another protocol without random oracles were proposed in 2001 by Katz, Ostrovsky, and Yung [16]. All those protocols are however at least as expensive as the Diffie-Hellman protocol.

Despite all this nice and extensive piece of theory, standards such as Bluetooth [1,2] stick to symmetric-key techniques (for cost reasons) and continue to use insecure protocols.

In this paper, we review the Bluetooth pairing protocol and its insecurity. The Bluetooth version 1.2 [1] mentioned (in a single sentence) the possibility to refresh keys. More details (namely, how to do so) were provided in Bluetooth version 2.0 in 2004 [2]. We finally show that this feature (that we call *repairing*) substantially increases the security and may be considered as a pragmatic costless solution. Security is based on the assumption that the radio channel (considered to be insecure by default) *sometimes* provides privacy in an unpredictable way, i.e. that the adversary Eve can in principle easily listen to the channel from time to time, but it is unlikely that she can do it *all the time* throughout the his-

tory of the devices association. This assumption is quite reasonable due to the mobility context of Bluetooth applications.

2 Bluetooth-like Pre-Pairing and the Security Issue

We assume a set of N possible participants with identifier strings ID_i. (Note that the notion of identity is rather weak since authentication will be based on a human user manipulating physical devices: it can just be a mnemonic identifier like "laser printer", maybe extended by a MAC address.) We assume that they all manage a local database of (K_j, ID_j) pairs, meaning that the current private key to be used with participant ID_j is K_j. The goal of a pairing protocol between Alice of identity ID_A and Bob of identity ID_B is to create (or replace) an entry (K, ID_B) in the database of ID_A and an entry (K, ID_A) in the database of ID_B so that the key K is the same and private to both participants.

For cost reasons, nowadays wireless devices (e.g. Bluetooth devices) only use symmetric-key cryptographic protocols for establishing secure communications over insecure channels. When they connect to each other for the first time, they establish some initial private key materials K^i. Both devices, Alice and Bob, start with their identities ID_A and ID_B, pick some random numbers R_A^i and R_B^i. Additionally, a user types some random one-time private code π on both devices and both devices run a π-based authenticated key agreement protocol. When they prompt the user to type π, they may display a piece of the identifier strings (a mnemonic) for user-friendliness reasons. Due to the state of the art on symmetric-key primitives, the protocol must leak R_A^i and R_B^i so that we have

$$K^i = G(\mathsf{ID}_A, \mathsf{ID}_B, R_A^i, R_B^i, \pi)$$

for some function G. In a one-move variant, R_B^i is void so that only R_A^i (which is rather denoted R^i) needs to be sent. (See Fig. 1.)[1]

Following our setting model, π has low entropy. Indeed, the private code is typed by a human user and is typically pretty small. Eventually, exhaustive search leads to guessing π. Hence, an adversary can typically compute K^i from R^i by guessing π. The adversary only needs some information about K^i to check whether π is correct or not to run an *offline* dictionary attack. Peer authentication protocols based on K^i are based on symmetric-key cryptography. They eventually leak such an information by releasing some S and $F(S, K^i)$ for some function F from the protocol. In the Bluetooth case, this attack was described by Jakobsson and Wetzel [17].

[1] By convention, notations without a hat are sent values and notations with a hat are received values. If no attack occurs, the value should not be changed by putting a hat.

This attack can be completed by a man-in-the-middle attack. Namely, an adversary can claim to have identity ID_B to Alice of identity ID_A and to have identity ID_A to Bob of identity ID_B. Even though the adversary does not get π from the user who wants to pair the real Alice and Bob, the adversary can easily infer it from the previous attack. The consequence is that Alice and Bob would be independently paired with the adversary even though they think they are paired together.

Those protocols can nevertheless be secure *in principle* provided that

- either enumerating all possible values for the code π is infeasible
- or the transmission of R^i is confidential.

In Section 6 we prove it in the random oracle model.

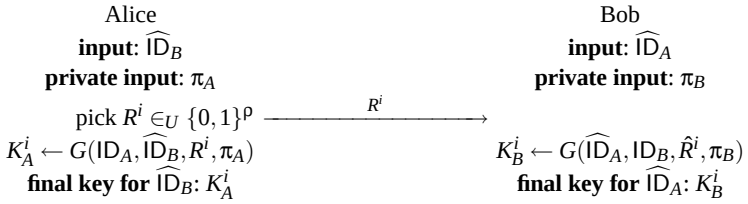

Fig. 1. A One-Move Preparing Protocol.

3 The Two-Round Bluetooth Pairing

The Bluetooth standard [1,2] is quite interesting in the sense that it uses a 2-round pairing protocol that we call *preparing* and *repairing*. Fig. 1 and Fig. 2 illustrate the two rounds, respectively. In a first round, a 128-bit (ephemeral) initialization key K^i is established from some random numbers R^i and π. In a second round, the final key is established from new random numbers R_A and R_B, the identities of Alice and Bob, and K^i. More precisely, the second round works as follows.

1. Bob picks a random R_B and sends $C_B = R_B \oplus K^i$ to Alice.
2. Alice picks a random R_A and sends $C_A = R_A \oplus K^i$ to Bob.[2]
3. Both compute $K = H(\mathsf{ID}_A, \mathsf{ID}_B, R_A, R_B) = H(\mathsf{ID}_A, \mathsf{ID}_B, C_A \oplus K^i, C_B \oplus K^i)$.

We assume that (K, ID_B) (resp. (K, ID_A)) replaces (K^i, ID_B) (resp. (K^i, ID_A)) in the database of ID_A (resp. ID_B) so that K^i is discarded.

[2] It is worth noticing that Alice and Bob actually exchange R_A and R_B by using a (safe) two-time pad.

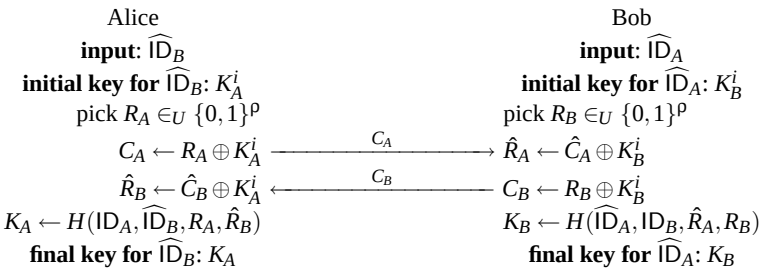

Fig. 2. The Bluetooth Repairing Protocol.

Note that the internal structure of H in Bluetooth is of the form

$$H(\mathsf{ID}_A, \mathsf{ID}_B, R_A, R_B) = H'(\mathsf{ID}_A, R_A) \oplus H'(\mathsf{ID}_B, R_B).$$

Obviously, this does *not* instantiate a random oracle since we have unexpected relations such as

$$H(\mathsf{ID}_A, \mathsf{ID}_B, R_A, R_B) \oplus H(\mathsf{ID}_B, \mathsf{ID}_C, R_B, R_C) = H(\mathsf{ID}_A, \mathsf{ID}_C, R_A, R_C).$$

We further note that if Alice and Bob were already the victims of a man-in-the-middle attack, they can remain in the same attacked state if the adversary can continue an active attack. When the adversary becomes out of reach, the repairing protocol fails and Alice and Bob end in a state so that they can no longer communicate.

In Section 6 we prove that the repairing protocol alone is secure if either the initialization key is private or the communication of either C_A or C_B is private. We deduce that the preparing and repairing together achieve a secure pairing protocol provided that either π is large or the communication is private: repairing does not decrease the security. The incremental role of the repairing protocol will be made clear in the following section.

4 Repairing and Forward Secrecy

The Bluetooth standard [1,2] already suggests that a key K could be refreshed. Indeed, new pairing protocols could just skip the first round and use the old K as the K^i initialization key. (Note that the user no longer needs to type a private code in this protocol.) If the old key was not known by the adversary, it could not be guessed like π. So the new link key would be safe as well. Now, if the old key K had leaked out, but the adversary did not listen to the new pairing protocol, then the new key would be safe: the secure communication would be

repaired. This way, we claim that the new link key is at least as safe as the old one.

Similarly, mobility and repairing can detect man-in-the-middle attacks as previously discussed. This repairs the weak notion of authentication.

Furthermore, frequent repairs provides *forward secrecy* when we make sure that old link keys are destroyed. Indeed, if we let K_j denote the link key generated by the jth pairing protocol, assuming that this pairing was safe and that K_{j+t} is the first key which is leaked after the jth pairing, then none of the link keys $K_j, K_{j+1}, \ldots, K_{j+t-1}$ can be recovered by the adversary. In the mobility context of Bluetooth, it is reasonable that the adversary does not listen to *all* pairing protocols. Since security only increases here, communications are eventually secure between Alice and Bob. It is indeed the case where mobility can help security.

What can happen in the case of active attacks? The two devices will end up in an unpaired state. Due to the mobility and the inability for the adversary to follow both devices, the user will eventually realize that the two devices are not paired and launch a new pairing protocol. An adversary could use this behavior and try a denial of service attack combined with social engineering: indeed the adversary could make sure that the two devices are unable to communicate, making as if the two devices were not well paired. The consequence would be for the user to launch a new pairing protocol with a humanly selected π. This would clearly provide openings to the adversary. This problem can only be fixed by clear human-machine interfaces and education of users. A pairing should not be perceived a benign action.

Another helpful feature would be, if possible, to enlarge the database by adding a new field telling the length of π in the preparing protocol and the number of repairs. Keys with low length for π and low number of repairs would have a low security confidence, but would become more trustable as the number of repairs increase.

To conclude, we believe that the repairing protocols provide a truly pragmatic and costless security solution for lightweight wireless communications.

5 Adversarial Model

The launch *and* send *oracles.* We adapt the security model of [5,6] and [22]. We assume that the powerful adversary can launch instances of the preparing/repairing protocol on chosen inputs by making a chosen participant to play the (chosen) role of Alice or Bob with a chosen input. For instance, the $\Pi \leftarrow \mathsf{launch}(n, \mathrm{Alice}, \mathsf{ID})$ query creates an instance π of Alice with input ID, played by node n. We assume that the adversary can play with all participants in a con-

current way and basically run the protocol step by step. The adversary is the central node of the communication channels, can send an arbitrary message to any instance and get the response message in return. For instance, $y \leftarrow \mathsf{send}(\Pi, x)$ sends the message x as being the current protocol message to instance Π, makes this instance step the protocol, and tells the protocol answer y of Π.

The test *oracle.* We assume that the adversary can make $\mathsf{test}(n, k, \mathsf{ID})$ oracle calls which tell whether (k, ID) is an entry of the database of node n. We say that an adversary wins if one test query positively answered. Note that contrarily to the traditional Bellare-Rogaway [5,6] model, the adversary can make as many test queries as he wants. The reason is that, in practice, information leaks so that the adversary can simulate this oracle in an offline way.

Every key K in a database can be seen as a random variable. In that case, every (unsuccessful) test query reduced the entropy by telling the adversary that K is not equal to a given k.

The remove *oracle.* We also assume that the adversary can make $\mathsf{remove}(n, \mathsf{ID})$ oracle queries which make node n remove any entry with ID from its database. This simulates a user managing the database of paired devices.

The inputPIN *oracle.* The preparing protocol assumes a special channel which privately sends the same random value π to two instances. We model this by the ability for the adversary to make some $\sigma \leftarrow \mathsf{inputPIN}(n_1)$ and $\mathsf{inputPIN}(\sigma, n_2)$ oracle calls which make n_1 receive a new random input π attached to a fresh tag σ, then n_2 receive the same input π. We assume that π is discarded by $\mathsf{inputPIN}(\sigma, n_2)$ after that (namely, a single π cannot be input more than twice). The distribution of π (namely, the entropy) will play a role in the analysis.

The reveal *and* corrupt *oracles.* Participating nodes are assumed to be honest by default. In the traditional Bellare-Rogaway model [5,6], the adversary can nevertheless make $\mathsf{reveal}(n)$ queries which simply dump the private database of a node n, and $\mathsf{corrupt}(n, \mathsf{code})$ queries which go beyond that by further installing a malicious code on the node so that this node can no longer be assumed to follow the protocols. For simplicity reasons we assume that adversaries have no access to these oracles here. Extensions of our results is left to further work. Note that excluding malicious participants de facto exclude the adversary from getting any π form inputPIN.

The secureLaunch *oracle.* The repairing protocol assumes that communication between two prepared participants can sometimes be private. Additionally, we sometimes consider instances of the preparing protocol that are also

run in a private environment. In such a case, we assume that an oracle query $\mathsf{secureLaunch}(n_A, n_B, x_A, x_B)$ launches a complete run of the protocol on nodes n_A and n_B with input x_A and x_B respectively. The adversary has no access to the transcript of the protocol.

6 Security of the Preparing Protocol

Theorem 1. *Given an integer* ρ *and a random oracle* G *which outputs* u*-bit strings, we consider the preparing protocol of Fig. 1. We assume that* $\mathsf{inputPIN}$ *selects* π *uniformly distributed in a set of* S *elements. For any adversary limited to* t test *queries, the wining probability is at most* $t/S + \frac{1}{2}t^2 2^{-u}$.

A key that was set up by $\mathsf{secureLaunch}$ *can only be successfully tested with probability at most* $\min(2^{-\rho}, 2^{-u})$.

We can easily tune u so that $t^2 \ll 2^u$. This result thus tells us that the protocol is secure when S is large. Typically, for $S = 2^{80}$ and $u = 128$, an adversary requires at least nearly 2^{64} trials so succeed. The theorem also says that if ρ and u are large and R^i is privately sent, then the protocol is secure.

Proof. Let us consider the ith test query $\mathsf{test}(n_i, k_i, \mathsf{ID}'_i)$ and assume that all previous test queries were negative. We want to compute the probability that the answer is positive. Due to the protocol, it relates to some random variable $K_i = G(\mathsf{ID}_i, \mathsf{ID}'_i, R_i, \pi_i)$ where R_i is known but π_i is a priori not.

Let L be the number of pairwise different $(\mathsf{ID}_i, \mathsf{ID}'_i, R_i)$ triplets. Let s_ℓ be the number of occurrences for the ℓth triplet, for $\ell = 1, \ldots, L$. Since G is a random oracle, it produces no collision $G(\mathsf{ID}_i, \mathsf{ID}'_i, R_i, \alpha) = G(\mathsf{ID}_i, \mathsf{ID}'_i, R_i, \beta)$ with probability higher than $1 - \frac{1}{2}s_\ell^2 2^{-u}$ where ℓ is the number of the triplet for the ith test. Let us focus in this case.

Clearly, the protocol leaks no information about any π, so information only comes from previous test oracles. Since G is a random oracle, any previous test query (let say the jth one) leaks some useful information about K_i only if $(\mathsf{ID}_j, \mathsf{ID}'_j, R_j, \pi_j) = (\mathsf{ID}_i, \mathsf{ID}'_i, R_i, \pi_i)$. Hence, the maximal information is that K_i is one value out of $S - s_\ell + 1$. The wining probability for this query is thus at most $1/(S - s_\ell + 1)$. The loosing probability for all queries related to this triplet is thus $1 - s_\ell/S$.

The overall loosing probability is thus at least

$$\prod_\ell \frac{S - s_\ell}{S} - \frac{1}{2}\sum_\ell s_\ell^2 2^{-u}$$

with constraint $\sum_\ell s_\ell = t$. The probability is the lowest for $L = 1$ for which it is $1 - t/S - \frac{1}{2}t^2 2^{-u}$.

When a key was set up by secureLaunch, we can best assume that the adversary caught π but no other information leaked. The best strategy to guess K^i is either to guess K^i or to guess R^i. with probability at most $\min(2^{-\rho}, 2^{-u})$. □

We similarly prove the following result.

Theorem 2. *Given an integer* ρ *and a random oracle H which outputs u-bit strings, we consider the preparing protocol of Fig. 2. We assume that initialization keys are randomly preset. For any adversary limited to t* test *queries, the wining probability is at most* $t^2 2^{-u}$.

A key that was repaired by secureLaunch *can only be successfully tested with probability at most* $\min(2^{-\rho}, 2^{-u})$.

7 Conclusion

We have shown that the pairing concept of Bluetooth can in principle lead to a secure protocol, provided that repairing is frequently done and is eventually privately run. This is proven provided that *G* and *H* behave like random oracles. This provides a pragmatic costless alternative to key agreement based on public-key cryptography.

We also proposed to store the length of the used PIN in the preparing protocol and the number of performed repairs in order to better assess the security of a given link key. This could help audit and increase the confidence in the Bluetooth security.

One open question would be to extend this result to the specific structure of the Bluetooth primitives. Another challenge would be to consider (namely to model and prove) security when the adversary has access to reveal or corrupt oracles.

References

1. Specification of the Bluetooth System. Vol. 2: Core System Package. Bluetooth Specification version 1.2, 2003.
2. Specification of the Bluetooth System. Bluetooth Specification version 2.0, 2004.
3. M. Abdalla, O. Chevassut, D. Pointcheval. One-Time Verifier-Based Encrypted Key Exchange. In *Public Key Cryptography'05*, Les Diablerets, Switzerland, Lecture Notes in Computer Science 3386, pp. 47–64, Springer-Verlag, 2005.
4. M. Bellare, D. Pointcheval, P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. In *Advances in Cryptology EUROCRYPT'00*, Brugge, Belgium, Lecture Notes in Computer Science 1807, pp. 139–155, Springer-Verlag, 2000.
5. M. Bellare, P. Rogaway. Entity Authentication and Key Distribution. In *Advances in Cryptology CRYPTO'93*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 773, pp. 232–249, Springer-Verlag, 1994.

6. M. Bellare, P. Rogaway. Provably Secure Session Key Distribution: the Three Party Case. In *Proceedings of the 27th ACM Symposium on Theory of Computing*, Las Vegas, Nevada, U.S.A., pp. 57–66, ACM Press, 1995.
7. M. Bellare, P. Rogaway. The AuthA Protocol for Password-Based Authenticated Key Exchange. In *Contribution to the IEEE P1363 study group for Future PKC Standards*, 2002. (Available from http://grouper.ieee.org/groups/1363/)
8. S. M. Bellovin, M. Merritt. Encrypted Key Exchange: Password-Based Protocols Secure Against Dictionary Attacks. In *IEEE symposium on Research in Security and Privacy*, Oakland, California, USA, pp. IEEE Computer Society Press, 72–84, 1992.
9. E. Biham, R. Chen, A. Joux, P. Carribault, C. Lemuet, W. Jalby. Collisions of SHA-0 and Reduced SHA-1. In *Advances in Cryptology EUROCRYPT'05*, Aarhus, Denmark, Lecture Notes in Computer Science 3494, pp. 36–57, Springer-Verlag, 2005.
10. V. Boyko, P. MacKenzie, S. Patel. Provably Secure Password Authenticated Key Exchange Using Diffie-Hellman. In *Advances in Cryptology EUROCRYPT'00*, Brugge, Belgium, Lecture Notes in Computer Science 1807, pp. 156–171, Springer-Verlag, 2000.
11. M. Čagalj, S. Čapkun, J.-P. Hubaux. Key Agreement in Peer-to-Peer Wireless Networks. To appear in the Proceedings of the IEEE, late 2005.
12. W. Diffie, M. E. Hellman. New Directions in Cryptography. *IEEE Transactions on Information Theory*, vol. IT-22, pp. 644–654, 1976.
13. C. Gehrmann, C. Mitchell, K. Nyberg. Manual Authentication for Wireless Devices. *RSA Cryptobytes*, vol. 7, pp. 29–37, 2004.
14. C. Gehrmann, K. Nyberg. Security in Personal Area Networks. In *Security for Mobility*, C. Mitchell (Ed.), pp. 191–230, IEE, 2004.
15. J.-H. Hoepman. The Ephemeral Pairing Problem. In *Financial Cryptography, 8th International Conference (FC 2004)*, Key West, Florida, USA, Lecture Notes in Computer Science 3110, pp. 212–226, Springer-Verlag, 2004.
16. J. Katz, R. Ostrovsky, M. Yung. Efficient Password-Authenticated Key Exchange using Human-Memorable Passwords. In *Advances in Cryptology EUROCRYPT'01*, Innsbruck, Austria, Lecture Notes in Computer Science 2045, pp. 475–494, Springer-Verlag, 2001.
17. M. Jakobsson, S. Wetzel. Security Weaknesses in Bluetooth. In *Topics in Cryptology (CT–RSA'01)*, San Francisco, California, USA, Lecture Notes in Computer Science 2020, pp. 176–191, Springer-Verlag, 2001.
18. P. MacKenzie. The PAK Suite: Protocols for Password-Authenticated Key Exchange. Technical report No. 2002-46. DIMACS Center, Rutgers University, 2002. (Available from http://dimacs.rutgers.edu/TechnicalReports/abstracts/2002/2002-46.html)
19. R. C. Merkle. Secure Communications over Insecure Channels. *Communications of the ACM*, vol. 21, pp. 294–299, 1978.
20. S. Pasini, S. Vaudenay. An Optimal Non-Interactive Message Authentication Protocol. To appear in the proceedings of CT-RSA'06, Springer, LNCS, 2006.
21. R. L. Rivest. The MD5 Message Digest Algorithm. RFC 1321, 1992.
22. S. Vaudenay. Secure Communications over Insecure Channels Based on Short Authenticated Strings. In *Advances in Cryptology CRYPTO'05*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 3621, pp. 309–326, Springer-Verlag, 2005.
23. X. Wang, X. Lai, D. Feng, H. Chen, X. Yu. Cryptanalysis for Hash Functions MD4 and RIPEMD. In *Advances in Cryptology EUROCRYPT'05*, Aarhus, Denmark, Lecture Notes in Computer Science 3494, pp. 1–18, Springer-Verlag, 2005.
24. X. Wang, H. Yu, L. Y. Yin. Efficient Collision Search Attacks on SHA-0. In *Advances in Cryptology CRYPTO'05*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 3621, pp. 1–16, Springer-Verlag, 2005.

25. X. Wang, L. Y. Yin, H. Yu. Finding Collisions in the Full SHA-1. In *Advances in Cryptology CRYPTO'05*, Santa Barbara, California, U.S.A., Lecture Notes in Computer Science 3621, pp. 17–36, Springer-Verlag, 2005.
26. X. Wang, H. Yu. How to Break MD5 and Other Hash Functions. In *Advances in Cryptology EUROCRYPT'05*, Aarhus, Denmark, Lecture Notes in Computer Science 3494, pp. 19–35, Springer-Verlag, 2005.
27. T. Wu. The Secure Remote Password Protocol. In *Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, San Diego, CA, pp. 97–111, The Internet Society, 1998.
28. T. Wu. The SRP Authentication and Key Exchange System. RFC 2945 standard track, The Internet Society, 2000.