Bit-Free Collision: Application to APOP Attack

Lei Wang[1], Yu Sasaki[1,2], Kazuo Sakiyama[1], and Kazuo Ohta[1]

[1] The University of Electro-Communications,
1-5-1 Chofugaoka, Chofu-shi, Tokyo, 182-8585, Japan
{wanglei,yu339,saki,ota}@ice.uec.ac.jp
[2] NTT Information Sharing Platform Laboratories, NTT Corporation
sasaki.yu@lab.ntt.co.jp

Abstract. This paper proposes a new variant of collisions on hash functions named *bit-free collision*, which can be applied to reduce the number of chosen challenges in password recovery attacks on hash-based challenge and response protocols, such as APOP (Authentication Post Office Protocol). In all previous APOP attacks, the attacker needs to impersonate the server and to send poisoned chosen challenges to the user. Impersonating the server takes a risk that the user may find out he is being attacked. Hence, it is important for the attacker to reduce the number of impersonation in order to lower the probability that the attack will be detected. To achieve this, reducing the number of chosen challenges is necessary. This paper is the first approach to improve previous APOP attacks based on this observation to our best knowledge. With t-bit-free collisions presented in this paper, the number of chosen challenges to recover each password character can be reduced by approximately a factor of 2^t. Though our attack utilizing t-bit-free collisions needs higher offline complexity than previous attacks, the offline computation can be finished in practical time if the attacker can obtain reasonable computation power. In this research, we generate 1-bit-free collisions on MD5 practically. As a result, the number of challenges for password recovery attacks on real APOP is approximately *half* reduced. Of independent interest, we apply the bit-free-collision attack on a simpler hash function MD4, and show that 3-bit-free collisions can be generated practically.

Keywords: hash function, bit-free collision, APOP, MD5, MD4.

1 Introduction

With the development of internet, challenge and response password authentication protocols have become popular. In the communication through internet, the user may face several threats: (a) a third party may impersonate the server, and (b) a third party may eavesdrop on the communication channel. Accordingly, it is dangerous for the user to send the password itself directly to the server to get authenticated. As a countermeasure to protect the password, challenge and response password authentication protocols have been adopted. The crucial idea is, in every authentication round, randomly generating challenges and computing responses based on challenges and the password. One popular approach to

T. Takagi and M. Mambo (Eds.): IWSEC 2009, LNCS 5824, pp. 3–21, 2009.

generate responses is hashing challenges and the password, which will base the security of the protocols on the underlying hash functions. As far as the hash function is secure, the protocol is secure.

APOP (Authentication Post Office Protocol) [8] is a challenge and response password authentication protocol based on MD5 [10], which has been *practically* utilized in real mail systems. The responses are generated by hashing the challenges concatenated with the password. Recently password recovery attacks on APOP have been proposed [6] [11] [12], which originated from collision attacks on MD5 [15]. All previous APOP attacks are *chosen* challenge attacks. The attacker impersonates the server and sends chosen challenges to the user to make the user provide corresponding responses. The attack scenario is as follows.

1. the user sends access requests to the attacker (impersonating the server).
2. the attacker sends poisoned chosen challenges to the user.
3. the user sends the corresponding responses to the attacker.
4. the attacker responds "No new email" to the user.

Impersonating the server takes a risk that the user may suspect to have being attacked in the following situations:

(a) the user does not get a new email for a long time if the attacker continuously impersonates the server.
(b) the user gets a new email delayed even if the attacker impersonates the server from time to time. Suppose the user accesses to the server once an hour. The user may get a new email delayed an hour even though the attacker only impersonates once.

From this observation, the attacker should reduce the times of impersonating in order to lower the probability that the attack will be detected. This means that the number of necessary chosen challenges should be reduced. So far, no result has been published to improve previous APOP attacks on reducing the number of chosen challenges to our best knowledge.

1.1 Our Results

We will propose the first approach to reduce the number of challenges of APOP attacks, by presenting a new variant of collisions on hash functions named *bit-free collision*.

Conceptually, bit-free collision is a pair of partially-fixed messages, which will collide regardless of the value of the unfixed bits. The unfixed bits are denoted as *free bits* in this paper. We denote by t-bit-free collision a bit-free collision with t-free bits for simplicity. For example, suppose (M, M') is a 1-bit-free collision. By setting the free bit to 0, we can obtain a pair of messages (M_0, M'_0) from (M, M'). By setting the free bit to 1, we can obtain another pair of messages (M_1, M'_1) from (M, M'). Both (M_0, M'_0) and (M_1, M'_1) are collisions. So (M, M') will be a collision no matter what the value of the free bit is. Similarly, for t-bit-free collision, a set of 2^t pairs of messages can be derived by setting the

values of the t-free bits, and each pair of messages from this set is a collision. Here we will roughly point out the difference between free bit in this paper and neutral bit proposed by Biham and Chen in [1]. The usage of free bit is mainly allowing the attacker to freely determine several bit-values of the colliding messages without affecting the collision, while the usage of neutral bit is mainly speeding up collision search. Refer to Section 3 for more details.

Bit-free collisions can be utilized to reduce the number of chosen challenges of APOP attacks. Previous APOP attacks recover the password characters one by one. One character is 8-bit long, so there are 2^8 possible candidates for one password character. The attacker adopts *guess-then-verify* approach to exhaustively check the correctness of all possible candidates for one password character. In order to check one possible candidate, the attacker needs to generate a pair of challenges. If a possible candidate is the *true* password character, the responses of the corresponding pair of challenges will collide. There are in total 2^8 candidates for a password character, and each candidate needs a pair of challenges. As a result, $(2^8 - 1)$ pairs of challenges[1] are necessary in the worst case.

On the other hand, our attack utilizes bit-free collisions to reduce the number of chosen challenges. Our attack will also recover password characters one by one. *The main novelty of our attack is recovering one password character part by part, which will reduce the number of necessary chosen challenges.* The high-level description of our attack to recover one password character is as follows. More details are explained in Section 5.2.

1. Locate t-free bits in the targeted password character, which divides the password character into $(8-t)$-non-free bits and t-free bits.
2. Recover the $(8-t)$-non-free bits first. The attacker adopts exhaustive guess-then-verify approach: guess the value of $(8-t)$-non-free bits, then generate a pair of challenges, which will lead to a t-bit-free collision after being concatenated with the guess value, and finally send the pair of challenges to the user to check whether the responses collide or not. If the responses collide, the guess value is true. Otherwise, the guess value is wrong. There are in total (2^{8-t}) possible candidates for the $(8-t)$-non-free bits, so $(2^{8-t}-1)$ pairs of challenges are necessary in the worst case.
3. Recover the t-free bits also by guess-then-verify approach. The attacker can recover the t-free bits one bit by one bit. For example, the attacker sets the value of the first target bit to 0, then generates a pair of challenges that leads to a $(t-1)$-bit-free collision, and finally sends the pair of challenges to check whether the responses collide or not. If the responses collide, then the guess is correct and the value of the first target bit is 0. Otherwise, the value of the first target bit is 1. Similarly, the attacker can recover the second bit until the whole t-free bits are obtained. So t pairs of challenges are necessary in the worst case.

In the above procedure of recovering one password character, the attacker will first exhaustively guess and recover the values of $(8-t)$-non-free bits without

[1] The attacker does not need to check the last possible candidate if the attacker has confirmed all other candidates are not correct.

the knowledge of the t-free bits, and then recover the value of the t-free bits. As shown above, $(2^{8-t} - 1 + t)$ pairs of challenges are necessary in the worst case. The total number of necessary challenges has been reduced by approximately a factor of 2^t compared to previous APOP attacks. Consequently following our attack strategy, the probability that the user will detect the attack has become lower, which makes the attack become more realistic.

We will analyze the complexity of generating bit-free collisions. In general, such bit-free collisions are harder to be found than regular collisions, but can provide more serious damages to hash-based protocols. We believe that bit-free collisions have more applications besides APOP attacks. We will use one compression function computation as a unit to count the complexity. The complexity of generating t-bit-free collisions on a general Merkle-Damgård hash function is $(\frac{n}{2})^{2^t} \times 2^{\frac{n}{2}+t}$ computations. Here we omit the descriptions. Refer to Section 3.1 for more details. The complexity of t-bit-free collisions on a general compression function is $2^{2^{t-1}\times n+t}$ computations. Refer to Section 3.2 for more details. Moreover, we apply bit-free-collision attacks on MD5 [10] and MD4 [9] by utilizing previous collision attacks on MD5 and MD4 [2] [15] [14]. We show that 1-bit-free-collision attacks on MD5 and 3-bit-free-collision attacks on MD4 are practical.

Finally, we show the effect of applying 1-bit-free collisions on MD5 to password recovery attacks on APOP in a real environment. The previous paper [6] assumes that each password character has 6 bits of entropy (they consider a kind of password most people use). Under this assumption, a password character is recovered by generating 2^5 colliding challenge pairs, and asking 2^6 queries impersonating the server. Previous paper [6] assumes that a user runs the authentication protocol once per minute, and estimates that asking 2^6 queries takes about 1 hour. In our attack, with the same assumption, we need to generate 2^4 1-bit-free collisions for recovering non-free bits and a single collision for recovering the free bit. Hence, our attack needs to ask 2^5+2 queries in the online phase, which can be finished in roughly 30 minutes[2]. Another our concern is the validity of the assumption that the attacker can ask chosen challenges once per minute. This assumption is not always true. It is typical that a user checks new mails only several times per day. Let us consider the case that the impersonation can be done only once per day. Clearly, the previous attack takes roughly 60 days to recover a password character while our attack takes only 30 days. We therefore can say that reducing the number of queries is important in a real environment.

1.2 Organization of the Paper

Section 2 explains background and related works. Section 3 defines bit-free collision and analyzes the complexity of generating bit-free collisions. Section 4 shows practical bit-free-collision attacks on MD5 and MD4 based on previous differential collision attacks. Section 5 applies bit-free collisions to APOP attack. Section 6 gives the conclusion and discussion of future work.

[2] We ignore the offline complexity of generating 1-bit-free collisions. Actually, this can be finished quickly, e.g. 36 seconds with 14400 PCs (details are discussed in Section 4.2). Hence, the assumption is reasonable.

2 Background and Related Works

2.1 Merkle-Damgård Hash Function

Many hash functions such as MD5 [10] and MD4 [9] have been designed following the well-known framework *Merkle-Damgård* [7] [3]. A Merkle-Damgård hash function map arbitrary-length messages to short hash digests by iterating a fixed-input-length component usually described as *compression function*. Denote by H, F and M a Merkle-Damgård hash function, underlying compression function and an input message respectively. The hash procedure is as follows:

1. M will be padded and divided into fixed-length blocks m_1, m_2, $\cdots$, m_l: $pad(M) = m_1||m_2||\cdots||m_l$, where $||$ means concatenation.
2. F takes a public constant IV and m_1 as input and outputs an intermediate value h_1. Then F takes h_1 and m_2 as input and outputs h_2. Similarly, the calculation will be carried out until all the message blocks are used.
3. Finally H outputs h_l as the hash digest.

We will describe one property of Merkle-Damgård hash functions, which has been adopted by APOP attacks.

One property of Merkle-Damgård hash function
Denote by M and M' two messages. $pad(M) = m_1||m_2||\cdots||m_l$ and $pad(M') = m'_1||m'_2||\cdots||m'_l$. Moreover there is some t $(1 \leq t \leq l)$ such that $m_i = m'_i$ $(\forall i : t \leq i \leq l)$. According to the above hash procedure, the following relation holds:

$$h_t = h'_t \Longrightarrow h_l = h'_l.$$

2.2 APOP

APOP is a hash-based challenge and response authentication protocol [8], which is used in mail system by servers to authenticate users. The procedure of APOP is detailed as follows. A mail server and a user share one common password.

1. The user sends one access request to the mail server.
2. The mail server generates a random challenge, and sends it to the user.
3. The user calculates one hash digest MD5(challenge||password), and sends the digest to the mail server.
4. The mail server itself carries out the same calculation, gets another hash digest, and compares it with the user's response.
5. If the two digests are the same, authentication succeeds. Otherwise, authentication fails.

2.3 Previous Password Recovery Attacks on APOP

Password recovery attacks on APOP [6] [11] [12] are chosen challenge attacks. The attacker impersonates the server and sends chosen challenges to the user. Briefly speaking, the attacker will recover the password characters one by one based on the property of MD5 (Merkle-Damgård hash function), which has been

shown in Section 2.1. Consequently, the complexity of recovering the whole password will be reduced significantly from the expected complexity. Denote the password by $P_1||P_2||\ldots||P_l$. Suppose the attacker has recovered the values of P_1, P_2, ..., P_{i-2} and P_{i-1} ($i \leq l$). The high-level description of the procedure of recovering P_i is as follows.

1. Guess the value of P_i.
2. Generate a pair of challenges (C, C') satisfying three conditions: C and C' have the same length; the length of $C||P_1||P_2||\cdots||P_i$ is multiple of block-length; and $H(C||P_1||P_2||\cdots||P_i) = H(C'||P_1||P_2||\cdots||P_i)$.
3. Send C to the user to obtain the response R.
4. Send C' to the user to obtain the response R'.
5. If $R = R'$, then the current guess value is the true P_i.
6. If $R \neq R'$, change the guess value, and go to step 2.

Suppose P_i has n bits. There are 2^n possible candidates for P_i. As a result, steps 2-6 will be repeated 2^n times in the worst case. So the bit-length of P_i should be as short as possible. From the specification of APOP [8], the length of challenges must be a multiple of 8 bits. Therefore, the minimum length of P_i is 8 bits, namely, one character. So, the previous APOP attacks [6] [11] [12] recover the password characters one by one. A password character is 8-bit long, so there are in total 2^8 possible candidates for one password character. Following the above previous APOP attack procedure, $2^8 - 1$ pairs of challenges are necessary in the worst case. This paper will mainly deal with how to reduce the number of necessary challenges.

3 Bit-Free Collision

Definition 1. *If a pair of partially-fixed messages (M, M') satisfies the following conditions*[3]*, it is denoted as a bit-free collision on a hash function H:*

1. *M and M' have the same bit-length.*
2. *M and M' have the unfixed bits at the same bit positions.*
3. *the unfixed bits of M and the unfixed bits of M' are equal.*
4. *any pair of messages, derived by setting the value of the unfixed-bits of M and M', will be a collision on H.*

where the unfixed bits are denoted as free bits.

Denote by t-bit-free collision a bit-free collision with t-free bits for simplicity. 2^t pairs of colliding messages can be derived from a t-bit-free collision. So a t-bit-free collision is a set of 2^t independent colliding message pairs.

Picking 1-bit-free collision, denoted as (M, M'), as an example. M and M' have the same bit-length, and have one same bit position (the free bit), where the value is not fixed. By setting the free bit to 0, a pair of messages (M_0, M'_0) is

[3] The conditions are restrictive. In fact we can give more general definition for bit-free collision. For example, conditions 1, 2 and 3 are not necessary. Since this paper deals with application to APOP attacks, we define the bit-free collision according to this application for consistency.

derived from (M, M'). By setting the free bit to 1, a pair of messages (M_1, M'_1) is derived from (M, M'). Both (M_0, M'_0) and (M_1, M'_1) are collisions on H.

This paper will deal with bit-free-collision attacks on the compression function of MD5, which makes this concept similar to the neutral bit concept proposed by Biham and Chen [1]. The concept of the neutral bit has been detailed in Appendix A. Free bit can be regarded as neutral bit up to the last step of MD5 compression function. Previous works never consider neutral bit up to the last step. This is because the usage of neutral bits is mainly speeding up the collision search. So the previous works are interested in finding neutral bits up to *some* intermediate step of hash computation. Free bit does not speed up the collision search. However, it has the following potential advantage: the attacker has the power to freely control *some* bit-values of the colliding messages without affecting the collision. We believe the concept bit-free collision has many applications.

In the following two sections, we will analyze the complexity of generating a t-bit-free collision on a general iterated hash function and a general compression function.

3.1 Bit-Free Collisions on a General Merkle-Damgård Hash Function

Denote by H a general Merkle-Damgård hash function. Denote by n the bit-length of hash values of H. We will utilize Joux's multi-collisions [4] to generate bit-free collisions on H. The format of generated bit-free-colliding messages with l-block length is as follows: $(M = m_0||m_1||\dots||m_{l-1}, M' = m_0||m'_1||\dots||m'_{l-1})$, and the free bits locate in the m_0.

To warm up, we first show how to generate 1-bit-free collisions. We will locate the free bits in the first message block m_0. m_0^0 and m_0^1 are derived by setting the 1-free bit of m_0 to 0 and 1, respectively.

1. Determine the bit position of the 1-free bit.
2. Set the 1-free bit to 0, and adopt Joux's multi-collision technique [4] to obtain $2^{\frac{n}{2}}$ multi-collisions on H as shown in Fig. 1 with $l = \frac{n}{2}$. The colliding messages will be denoted as $m_0^0||m_{1,k_1}||\dots||m_{\frac{n}{2},k_{\frac{n}{2}}}$, where $k_1, k_2, \dots, k_{\frac{n}{2}} \in \{0, 1\}$.
3. Calculate the hash values of messages $m_0^1||m_{1,k_1}||m_{2,k_2}||\dots||m_{\frac{n}{2},k_{\frac{n}{2}}}$, for all $k_1, k_2, \dots, k_{\frac{n}{2}}$.

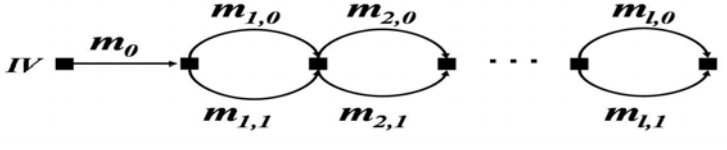

Fig. 1. Joux's multi-collision

4. If a pair of colliding messages is found: $(m_0^1||m_{1,k_1}||m_{2,k_2}||\ldots||m_{\frac{n}{2},k_{\frac{n}{2}}}$, $m_0^1||m_{1,k_1'}||m_{2,k_2'}||\ldots||m_{\frac{n}{2},k_{\frac{n}{2}}'})$, where $k_1, k_1', \ldots, k_{\frac{n}{2}}, k_{\frac{n}{2}}' \in \{0,1\}$, then $(m_0||m_{1,k_1}||\ldots||m_{\frac{n}{2},k_{\frac{n}{2}}}$, $m_0||m_{1,k_1'}||\ldots||m_{\frac{n}{2},k_{\frac{n}{2}}'})$ is 1-bit-free collision.

Similarly, we can utilize Joux's multi-collision technique to generate t-bit-free collisions. The high-level description is as follows:

1. Determine the bit positions for the t-free bits in m_0. There are 2^t possible values for the t-free bits, which will be denoted as $\{0, 1, \ldots, 2^t - 1\}$ for simplicity.
2. Set the values of the t-free bits to 0.
3. Generate multi-collisions on H as shown in Fig. 1 with $l = (\frac{n}{2})^{2^t}$. The complexity is $(\frac{n}{2})^{2^t} \times 2^{\frac{n}{2}}$, counting one compression function computation as a unit.
4. Change the free bits to 1. Denote the new derived first message block as m_0^1.
5. First calculate the intermediate hash value at $(\frac{n}{2}+1)$-th block of new messages (starting with m_0^1) to find a collision $(M_{1,0}, M_{1,1})$. Denote the colliding intermediate hash value as h. Then fix the intermediate hash value at $(\frac{n}{2}+1)$-th block as h, and calculate the intermediate hash values at $(n+1)$-th block to find a collision $(M_{2,0}, M_{2,1})$. Similar calculation will be carried out until the last message block. Finally we obtain $m_0||M_{1,k_1}||M_{2,k_2}||\ldots||M_{(\frac{n}{2})^{2^t-1},k_{(\frac{n}{2})^{2^t-1}}}$ $(k_i \in \{0,1\})$, which will collide when the value of the free bits are 0 or 1.
6. Repeat steps 4 and 5 setting the free bits to the rest possible values.
7. Finally we will obtain a pair of messages, which can collide for any possible value of the free bits. This pair of message is a t-bit-free collision on H.

The complexity of generating a t-bit-free collision on H is roughly $(\frac{n}{2})^{2^t} \times 2^{\frac{n}{2}+t}$.

3.2 Bit-Free Collisions on a General Compression Function

Denote by F a general compression function. Denote by n the bit-length of the outputs of F. We will assume that the message space is always large enough to carry out the exhaustive t-bit-free collision search. We will analyze the complexity of generating a t-bit-free collision on F by the exhaustive search. The exhaustive search is as follows:

1. Determine the bit positions for the t-free bits. There are in total 2^t possible values for the whole t-free bits, which will be denoted as $\{0, 1, \ldots, 2^t - 1\}$.
2. Randomly select a message M, and expand M to a set of messages $\{M_0, \ldots, M_{2^t-1}\}$, where M_i differs from M only at the t-free bits and the value of the t-free bits is i.
3. Search a pair of messages (M, M') such that $F(M_i) = F(M_i')$ for any $i \in \{0, 1, \ldots 2^t - 1\}$.
4. (M, M') is a pair of t-bit-free collision on F.

Denote by F^* a compression function $F^*(M) = F(M_0)||F(M_1)||\ldots||F(M_{2^t-1})$. The exhaustive search can be regarded as searching a collision on F^*. The bit-length of F^* is $2^t \times n$. So the complexity of generating a t-bit-free collision on

a general compression function F with n bit-length hash digests is $2^{(2^t \times n)/2}$ F^* computations. One F^* computation consists of 2^t F computations. As a result, t-bit-free-collision attacks on F is with a complexity $2^{2^{t-1} \times n + t}$ F computations.

4 Bit-Free-Collision Attacks Based on Differential Collision Attacks on Hash Functions

This section will deal with how to find bit-free collisions based on Wang *et al.*'s differential collision attacks.

In 2005, Wang *et al.* published their differential collision attacks on hash functions from MD4 family [14] [15]. Here we will briefly recall Wang *et al.*'s collision attacks. Refer to Appendix C.1 for more details. The attack procedure is as follows: first determine a message difference Δ, then determine how the Δ will propagate during hash computation, which is usually denoted as *differential path*, then derive sufficient conditions that make sure the difference propagation will follow the differential path, and finally search a message M satisfying all sufficient conditions, which leads to a collision $(M, M + \Delta)$.

4.1 Crucial Ideas

We first show a 1-bit-free collision as an example in Fig. 2. Denote by Δ the message difference of the chosen collision attack on H in Fig. 2. Following Wang *et al.*'s collision attacks, if M^0 can satisfy all the sufficient conditions, $(M^0, M^0 + \Delta)$ is a collision. Similarly, if M^1 can satisfy all the sufficient conditions, $(M^1, M^1 + \Delta)$ is a collision. So, as long as both M^0 and M^1 can satisfy all the sufficient conditions, $(M, M + \Delta)$ will collide no matter what the value of the free bit is. Consequently, 1-bit-free-collision search can be transformed to search a pair of messages (M^0, M^1) satisfying two conditions: M^0 and M^1 only differ at the free bit, and both M^0 and

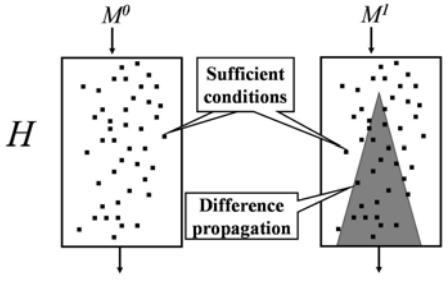

Fig. 2. a 1-bit-free-collision based on a collision attack

Sufficient conditions are from a collision attack on H. M^0 and M^1 are derived by setting a bit-value of M to 0 and 1, respectively. Difference propagation describes how the 1-bit difference between M^0 and M^1 propagates during the hash computation.

M^1 can satisfy all sufficient conditions of a differential path for collision attack. Usually the difference propagation in Fig. 2 starts from *some* intermediate step of hash computation depending on the bit position of the free bit. So after the bit positions of the free bits are determined, a suitable differential path should have the minimum number of sufficient conditions, which might be affected by the difference propagation in Fig. 2.

A high-level description of t-bit-free-collision attack based on Wang *et al.*'s collision attacks is as follows.

1. Pre-determine the bit positions of t-free bits.
2. Choose a differential path of a collision attack with the minimum number of sufficient conditions, which might be affected by changing the values of the free bits. Denote by Δ the message difference of the collision attack.
3. Search a message M that can satisfy all the sufficient conditions no matter what the value of the t-free bits is.
4. $(M, M + \Delta)$ is a t-bit-free collision.

4.2 Bit-Free-Collision Attacks on MD5

This section deals with bit-free-collision attacks on MD5. For specification of MD5, refer to Appendix B.

Pre-determine the bit position of the free bits:
Since we will apply bit-free-collision attacks on MD5 to improve APOP attacks, we locate the free bits in the last 8-bit positions of messages.

Well-suited differential path for collision attacks on MD5:
We adopt a differential path obtained from previous attacks on MD5 as follows:

-Pseudo-collision attack on the compression function of MD5 proposed by Den Boer and Bosselaers in [2] has been shown in Appendix C.2. The differential path for the pseudo-collision attack on MD5 has the least number of sufficient conditions to our best knowledge. Moreover, Klima in [5] proposes a fast collision search technique on MD5 named *Tunnel*. By using Klima's tunnel Q9 detailed in Appendix C.3, the bit-free collision search can be regarded as starting from step 25. So the number of sufficient conditions, which might be affected by changing the value of the free bits, will be only 23.

-A so-called technique *IV bridge* proposed by Sasaki *et al.* in [12] provides a pair of messages which can link public constant IV of MD5 to necessary differences for pseudo-collision attacks in [2]. The pair of messages in [12] is shown in Table 1: $(M_0||M_1, M_0||M_1')$, which provides a pair of intermediate hash values (h, h') well-suited for pseudo-collision attacks in [2].

Finally there are roughly 23 sufficient conditions that might be affected by changing the value of the free bits. For one collision on MD5, roughly 2^{23} MD5 computations are necessary. Hence, for t-bit-free collisions, $2^{23\times 2^t}$ MD5 computations are necessary. It seems that 1-bit-free collision with a complexity 2^{46} is practical.

Table 1. One example of 1-bit-free collision on MD5

$(M_0||M_1||M_2, M_0||M_1'||M_2)$ is a 1-bit-free collision, where the free bit locates at the 24-th bit of m_{15} of M_2. Denote by $md5$ the compression function of MD5.
$h = md5(md5(IV, M_0), M_1)$ and $h' = md5(md5(IV, M_0), M_1')$. h and h' is well-suited for pseudo-collision attacks on MD5 [2]: $md5(h, M_2) = md5(h', M_2)$.

M_0	m_4=0x37373936	m_5=0x3433302d	m_6=0x38312d35	m_7=0x61704035
	m_8=0x6f777373	m_9=0x645f6472	m_{10}=0x63657465	m_{11}=0x5f726f74
	m_{12}=0x2e636264	m_{13}=0x6976746d	m_{14}=0x632e7765	m_{15}=0x73752e61
M_1	m_0=0x986e1da4	m_1=0x83707d06	m_2=0xa86e1ddd	m_3=0xe264eedb
	m_4=0xff68e19f	m_5=0x120ea5b3	m_6=0x7437d3e2	m_7=0x600f543d
	m_8=0x7c63c5ab	m_9=0xe9ead9d9	m_{10}=0xa9b5c51e	m_{11}=0x**c**309f623
	m_{12}=0xfd534f1e	m_{13}=0xad33c7ad	m_{14}=0xfd0380c6	m_{15}=0x7745f36a
M_1'	m_0'=0x986e1da4	m_1'=0x83707d06	m_2'=0xa86e1ddd	m_3'=0xe264eedb
	m_4'=0xff68e19f	m_5'=0x120ea5b3	m_6'=0x7437d3e2	m_7'=0x600f543d
	m_8'=0x7c63c5ab	m_9'=0xe9ead9d9	m_{10}'=0xa9b5c51e	m_{11}'=0x**4**309f623
	m_{12}'=0xfd534f1e	m_{13}'=0xad33c7ad	m_{14}'=0xfd0380c6	m_{15}'=0x7745f36a
h	$h[a]$ = 0x**b**d7ade50;	$h[b]$ = 0x**e**17a619d;	$h[c]$ = 0x**8**e940937;	$h[d]$ = 0x**f**d4af95f;
h'	$h'[a]$ = 0x**3**d7ade50;	$h'[b]$ = 0x**6**17a619d;	$h'[c]$ = 0x**0**e940937;	$h'[d]$ = 0x**7**d4af95f;
M_2	m_0 = 0xc0797ae2;	m_1 = 0xe95d42e6;	m_2 = 0x49fe29af;	m_3 = 0x3329c9a9;
	m_4 = 0xa790a55d;	m_5 = 0x783e6d3;	m_6 = 0xb906c7b1;	m_7 = 0x2d63e951;
	m_8 = 0x9edac296;	m_9 = 0x26afe101;	m_{10} = 0xd4cfc4fb;	m_{11} = 0xcb0d1667;
	m_{12} = 0x77b75eab;	m_{13} = 0xea993a34;	m_{14} = 0x8c9868ae;	m_{15} = 0x7**e**ffffff;
	24-th bit of m_{15} of M_2 is free bit: m_{15} = 0x7effffff or 0x7fffffff.			

We implement 1-bit-free collision search. Surprisingly, it takes only 12 hours by 12 computers on average to generate 1-bit-free collision, which is much faster than the usual time for 2^{46} computations. One reason is that the complexity calculated by counting the number of sufficient conditions is greater than the precise complexity. Moreover, due to the biased bit position of sufficient conditions, since all sufficient conditions are located in only MSB of intermediate values, the complexity should be less than 2^{46}.

One example of generated 1-bit-free collision is shown in Table 1.

4.3 Bit-Free-Collision Attacks on MD4

We will also apply bit-free-collision attacks on MD4 [9]. For the specification of MD4, refer to Appendix B.

Pre-determine the bit position of the free bits:
Similarly with MD5 case, considering the application to APOP attack, we set the free bits at the last 8-bit positions of messages.

A well-suited differential path for collision attacks on MD4:
We determine to use the differential path on MD4 in [13] detailed in Appendix C.4, since it has the minimum number of sufficient conditions which might be affected by changing the value of the free bits to the best of our knowledge.

Table 2. One example of 3-bit-free collision on MD4

M_0	m_0 = 0x3938313c; m_1 = 0xbfdc10ea; m_2 = 0xc5708671; m_3 = 0xa0196be0; m_4 = 0xa8d2a83a; m_5 = 0xfd15dd85; m_6 = 0x992e75bc; m_7 = 0xabc6ccb8; m_8 = 0x6f6fd206; m_9 = 0xfd303797; m_{10} = 0x764081f6; m_{11} = 0xd6821ee2; m_{12} = 0xcc7e0ed5; m_{13} = 0x53c72d75; m_{14} = 0x446d4fe9; m_{15} = 0x1854dfdc;
M_1	m_0 = 0x182994f8; m_1 = 0xc989fe5e; m_2 = 0xe3e086f0; m_3 = 0x17eb1082; m_4 = 0x562a7af6; m_5 = 0xa6f0e339; m_6 = 0xc46682a8; m_7 = 0xb817cfa4; m_8 = 0xe5a24a72; m_9 = 0x8eca35be; m_{10} = 0x12c6229e; m_{11} = 0xaf84be49; m_{12} = 0x1a94a2a5; m_{13} = 0x8a2386b0; m_{14} = 0x76d2a8b1; m_{15} = 0x003effff;
	27-th, 28-th and 29-th bits of m_{15} of M_1 are the 3-free bits: m_{15}=0x003effff, 0x083effff, 0x103effff, 0x183effff, 0x203effff, 0x283effff, 0x303effff, 0x383effff

Here we will show one 3-bit-free collision example $(M_0||M_1, M_0||(M_1 + \Delta))$, where M_0 and M_1 have been shown in Table 2. The message difference ΔM $(M_1' - M_1)$ is $(\Delta m_0 = 2^{28}, \Delta m_2 = 2^{31}, \Delta m_4 = 2^{31}, \Delta m_8 = 2^{31}, \Delta m_{12} = 2^{31})$. No matter what the value of the 3-free bits (27, 28, 29-th bits of m_{15} of M_1) is, $\mathrm{MD4}(M_0||M_1) = \mathrm{MD4}(M_0||(M_1 + \Delta M))$.

5 Application to APOP Attacks

5.1 Overview of Our Contribution

Previous APOP attacks [6] [11] [12] are *chosen* challenge attacks. The attacker will impersonate the server and send chosen challenges to the user. Impersonating the server takes a risk that the user may suspect being attacked by the following situation: (a) the user does not get a new email for a long time if the attacker continuously impersonates the server; (b) the user get a new email delayed even when the attacker impersonates the server from time to time. To lower the probability that the attack will be detected, the number of impersonation should be reduced. To achieve this, the number of necessary challenges has to be reduced. This section will utilize bit-free collisions on MD5 to reduce the number of necessary chosen challenges. As shown in Section 4.2, 1-bit-free-collision attack on MD5 is practical. We will adopt 1-bit-free-collisions on MD5 to improve previous APOP attacks. The number of necessary challenges is almost *half*-reduced. For one password character, our attack needs 2^7 pairs of challenges in the worst case, while previous APOP attack needs $(2^8 - 1)$ pairs of challenges.

5.2 Improved APOP Attack

In this section, we will detail how to improve previous APOP attacks utilizing bit-free-collisions. Our attacks will also recover the password characters one by one, following previous attacks. Our attack procedure with a comparison with previous attacks has been shown in Table 3.

Table 3. Comparison between our attack and previous attacks

Denote by p_r the password characters which have been recovered. Denote by p a password character which is going to be recovered. Note that the online work and offline work are *parallel* and *independent*.

Our procedure

Our attack utilizes 1-bit-free collision. Set the bit position of the free bit in p, which will divide the p into two parts: 1-free bit and 7-non-free bits denoted as p_f and p_{nf}, respectively. For simplicity, we assume that the 1-free bit locates at MSB of p.

Stage 1: recover the value of p_{nf}.

-Chosen challenge collection (offline)

1. For $p_{nf} = 0000000$ to 1111111 (7-non-free bits)
2. Generate a pair of challenges (C, C'): $(C||p_r||p_f||p_{nf},\ C'||p_r||p_f||p_{nf})$ is a 1-bit-free collision.
3. Store (C, C', p_{nf}) to Table $\mathcal{T}$.
4. End For

-Impersonating as server (online)

If $\mathcal{T}$ is not NULL, then

1. Pick an element $(C,\ C',\ p_{nf})$ from $\mathcal{T}$.
2. Erase $(C,\ C',\ p_{nf})$ from $\mathcal{T}$.
3. Send C to the user to obtain the response R.
4. Send C' to the user to obtain the response R'.
5. If $R = R'$, then the current value p_{nf} is the true 7-non-free bits of p. Goto Stage 2.
6. If $R \neq R'$, continue to run Stage 1.

Else, the attacker does not impersonate. Continue to run Stage 1.

Stage 2: recover the value of p_f.

1. Guess the 1-free bit is 0.
2. Generate a pair of challenges (C, C') such that $(C||p_r||0||p_{nf},\ C'||p_r||0||p_{nf})$ is a collision.
3. Send C to the user to obtain the response R.
4. Send C' to the user to obtain the response R'.
5. If $R = R'$, the value of p_f is 0. Otherwise, the value is 1.
6. Halt the program.

Previous procedure

-Chosen challenge collection (offline)

1. For p=00000000 to 11111111 (8 bits)
2. Generate a pair of challenges (C, C'): $(C||p_r||p,\ C'||p_r||p)$ is a collision.
3. Store (C, C', p) to Table $\mathcal{T}$.
4. End For

-Impersonating as server (online)

If $\mathcal{T}$ is not NULL, then

1. Pick an element (C, C', p) from $\mathcal{T}$.
2. Erase (C, C', p) from $\mathcal{T}$
3. Send C to the user to obtain the response R.
4. Send C' to the user to obtain the response R'.
5. If $R = R'$, then the current value p is the target password character. Halt the program.
6. If $R \neq R'$, continue to run the program.

Else, the attacker does not impersonate. Continue to run the program.

As shown in Table 3, our procedure generates 2^7 chosen challenge pairs at Step 1 of the offline phase and 1 chosen challenge pair at Step 2 of Stage 2, whereas, the previous procedure generates 2^8 chosen challenge pairs at Step 1 of the offline phase. Hence, the number of chosen challenges in our attack is roughly half of the previous attack. Note our attack needs to generate 1-bit-free collisions. This requires higher complexity than generating collisions, but can be computed at offline. In the real protocol, it is typical that the protocol is triggered by the user, not by the server (or the attacker impersonating the server). Therefore, to make the user provide the responses of chosen challenges, the attacker needs to wait for the user's access requests. Such a waiting time might be long, e.g., half day. During this time, the attacker can process the offline part in parallel. Since 1-bit-free collisions of MD5 can be generated in 12 hours with 12 PCs as described in Section 4.2, we can conclude that the extra cost of offline complexity has less impact than reducing the number of chosen challenges in the real environment.

Application to APOP-MD4
Suppose APOP utilizes MD4 instead of MD5. As shown in Section 4.3, 3-bit-free collisions on MD4 can be found practically. For APOP-MD4 case, the attacker can adopt similar attack procedure with Table 3: first recover the non-free bits and then recover the free bits. So for one password character (8-bit long) in the worst case, $(2^{8-3}-1)$ pairs of challenges are necessary to recover the $(8-3)$-non-free bits, and 2^3 pairs of challenges are necessary to recover the 3-free bits. In total, the number of necessary challenges is 71 for one password character in the worst case, while previous attacks need 510 challenges. So the number of necessary challenges has been reduced by a factor of 7.2 compared with previous attacks.

6 Conclusion and Discussion

In this paper, we presented the first approach of reducing the number of chosen challenges in the APOP attacks. The newly proposed variant of collision "bit-free collision" enabled us to achieve this. Roughly speaking, when t-bit-free collisions are available, the number of chosen challenges becomes $1/2^t$ compared to the previous attacks. We showed how to generate t-bit-free collisions in general case, 1-bit-free collisions on MD5, and 3-bit-free collisions on MD4 with giving examples of generated bit-free collisions on MD5 and MD4. We applied bit-free collisions to APOP attacks, and proposed the improved attack procedure.

Finally we would like to discuss potential applications of bit-free collisions, which will be our future work. Here we will give one application on distinguishing a compression function family from a random function family. Moreover, all the elements of the compression function family share the same structure but differ in IV values. During interacting with the distinguisher, each of the two families changes its element to calculate responses from time to time. Suppose the attacker has enough offline computational power. The distinguishing attack procedure is as follows.

1. Locate the free bits in IV, then guess the bit-values of non-free bits of IV, and finally generate a t-bit-free collision (M, M') on the compression function.

2. Send M and M' to a oracle to obtain responses R and R' respectively.
3. If R is equal to R', then the oracle is the compression function family.

Denote by n the bit-length of IV. Suppose the distinguisher uses t-bit-free collisions. The bit-length of non-free bits is $(n-t)$, so if the oracle is the compression function family, after 2^{n-t} pairs of messages are queries, a pair of colliding responses will be obtained with non-negligible probability. On the other hand, a pair of colliding responses will be obtained after roughly 2^n pairs of messages are queried. Consequently, utilizing bit-free collisions, the distinguisher can succeed with non-negligible probability with an online rough complexity 2^{n-t+1} queries.

We expect more applications of bit-free collisions can be found in future.

References

1. Biham, E., Chen, R.: Near-collisions of SHA-0. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 290–305. Springer, Heidelberg (2004)
2. den Boer, B., Bosselaers, A.: Collisions for the Compression Function of MD-5. In: Helleseth, T. (ed.) EUROCRYPT 1993. LNCS, vol. 765, pp. 293–304. Springer, Heidelberg (1994)
3. Damgård, I.: A design principle for hash functions. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 416–427. Springer, Heidelberg (1990)
4. Joux, A.: Multicollisions in iterated hash functions. Application to cascaded constructions. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 306–316. Springer, Heidelberg (2004)
5. Klima, V.: Tunnels in Hash Functions: MD5 Collisions Within a Minute. Cryptology ePrint Archive, Report 2006 /105, `http://eprint.iacr.org/2006/105.pdf`
6. Leurent, G.: Message freedom in MD4 and MD5 collisions: Application to APOP. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 309–328. Springer, Heidelberg (2007)
7. Merkle, R.C.: One Way Hash Functions and DES. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 428–446. Springer, Heidelberg (1990)
8. Myers, J., Rose, M.: Post Office Protocol - Version 3. RFC 1939 (Standard), Updated by RFCs 1957, 2449 (May 1996), `ftp://ftp.isi.edu/in-notes/rfc1939.txt`
9. Rivest, R.L.: The MD4 Message Digest Algorithm. Request for Comments (RFC 1320), Network Working Group (1992)
10. Rivest, R.L.: The MD5 Message Digest Algorithm. Request for Comments (RFC 1321), Network Working Group (1992)
11. Sasaki, Y., Yamamoto, G., Aoki, K.: Practical Password Recovery on an MD5 Challenge and Response. Cryptology ePrint Archive, Report 2007/101
12. Sasaki, Y., Wang, L., Ohta, K., Kunihiro, N.: Security of MD5 challenge and response: Extension of APOP password recovery attack. In: Malkin, T.G. (ed.) CT-RSA 2008. LNCS, vol. 4964, pp. 1–18. Springer, Heidelberg (2008)
13. Sasaki, Y., Wang, L., Ohta, K., Kunihiro, N.: New message difference for MD4. In: Biryukov, A. (ed.) FSE 2007. LNCS, vol. 4593, pp. 329–348. Springer, Heidelberg (2007)
14. Wang, X., Lai, X., Feng, D., Chen, H., Yu, X.: Cryptanalysis of the hash functions MD4 and RIPEMD. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 1–18. Springer, Heidelberg (2005)
15. Wang, X., Yu, H.: How to break MD5 and other hash functions. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 19–35. Springer, Heidelberg (2005)

A Definition of Neutral Bits [1]

Biham and Chen proposed a concept *neutral bit* to speed up the collision search on hash functions. Here we will give a brief description. For more detailed description, refer to [1].

Definition 2. *[1] For a pair message M_0 and M_1, denote by Δ_i the difference of intermediate chaining variables at i-th step during hash computations. The j-th bit of M_0 and M_1 is a neutral bit with respect to M_0 and M_1 up to i-th step if it can satisfy the following property: M_0' and M_1' are obtained by flipping the j-th bit of M_0 and M_1, respectively, and the Δ_i' of M_0' and M_1' is equal to Δ_i of M_0 and M_1.*

B Specification of MD5 and MD4

MD5 [10] and MD4 [9] map arbitrary length messages to 128 bit-length hash digests. At first, the input message is padded and divided into 512-bit blocks. Here we will omit the description of padding rule. Then the message blocks will be sent to a primitive called *compression function* sequentially and hashed. A fixed 128-bit constant *initial value (IV)* and M_1 will be hashed by the compression function, which outputs a 128 bit-length H_1. Then H_1 and M_2 will be hashed by the compression function. After the last message block is hashed, the output of the compression function will be the hash digest.

In the following, we will briefly describe the compression functions of MD5 and MD4 respectively.

Compression function of MD5
The message block M and the intermediate value H will be divided into 32-bit values denoted as $(m_0, \ldots, m_{15})$ and (a_0, b_0, c_0, d_0) respectively. The compression function consists of 64 steps, regrouped into four 16-step rounds. Each step is defined as follows:

$$a_i = d_{i-1}, c_i = b_{i-1}, d_i = c_{i-1},$$
$$b_i = b_{i-1} + (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t) \lll s_i,$$

where m_k is one of $(m_0, \ldots, m_{15})$, the index k being given by a permutation of $\{0, \ldots, 15\}$ depending on the round, t is a constant defined in each round, $\lll s_i$ means a left-rotation by s_i bits, and f is a Boolean function depending on the round.

1R: $f(X, Y, Z) = (X \wedge Y) \vee (\neg X \wedge Z)$
2R: $f(X, Y, Z) = (X \wedge Z) \vee (Y \wedge \neg Z)$
3R: $f(X, Y, Z) = X \oplus Y \oplus Z$
4R: $f(X, Y, Z) = (X \vee \neg Z) \oplus Y$

The final output is $(a_0 + a_{64}, b_0 + b_{64}, c_0 + c_{64}, d_0 + d_{64})$.

Compression function of MD4
The differences between MD5 and MD4 are the following:

- MD4 consists of 48 steps regrouped into three 16-step rounds.
- Each step is defined as: $b_i = (a_{i-1} + f(b_{i-1}, c_{i-1}, d_{i-1}) + m_k + t) \lll s_i$, where m_k is given by different round permutations.
- In the 2nd round: $f(X, Y, Z) = (X \wedge Y) \vee (Y \wedge Z) \vee (X \wedge Z)$.

C Previous Related Collision Attacks on MD5 and MD4

C.1 Wang *et al.*'s Differential Collision Attack

Current popular collision attack on hash functions are mainly differential attacks following the strategy proposed by Wang *et al.* [15] [14]. Here we will describe the procedure of collision attacks.

1. Find the "Message Difference (ΔM)" that yields a collision with high probability. Let M and M' be a pair of messages that yield a collision. Difference ΔM is defined to be the value yielded by subtracting M from M': $\Delta M = M' - M$.
2. Determine how the impact of ΔM propagates. The propagation of the message difference at all intermediate statements is fixed and called the "Differential Path (DP)."
3. Derive "Sufficient Conditions (SC)" from differential path to guarantee that the message difference will propagate following the differential path at all intermediate statements.
4. Apply the technique called "Message Modification (MM)" such that a randomly selected message can be modified to make several sufficient conditions be satisfied.
5. Search a message that satisfies all SCs as follows: first randomly select a message, then modify it by message modification to make several sufficient conditions satisfied, and finally check whether the other sufficient conditions are satisfied or not. Denote the obtained message as M.
6. Calculate $M' = M + \Delta M$. M and M' will be a collision pair.

Complexity of collision attacks
In the above attack procedure, the first three steps are pre-stage works before searching collisions, and they are carried out only once. So the complexity of these three steps is not counted into the complexity of the collision attack. In steps 4 and 5, based on the technique MM, the SCs are divided into two cases: 1) the SCs can be satisfied by applying MM to any randomly selected message; 2) the SCs have to be satisfied by testing randomly selected messages, that is the exhaustive search. So the current popular approach of calculating the complexity of collision attacks is counting the number of the SCs of the second case. Denote the hash function as H. Suppose there are q SCs of the second case. Then the complexity of the collision attack is roughly regarded as 2^q H computations.

C.2 Pseudo-Collision Attacks on MD5

A pseudo-collision on the compression function of MD5 has been proposed by Den Boer and Bosselaers in [2], where the differences exist in the intermediate hash values instead of the message blocks. Denote the intermediate hash value as (a_0, b_0, c_0, d_0). The XOR differences are

$$(\Delta a_0, \Delta b_0, \Delta c_0, \Delta d_0) = (0x80000000, 0x80000000, 0x80000000, 0x80000000).$$

Moreover, An extra condition is that the MSBs of b_0, c_0 and d_0 should be equal. The sufficient conditions are as follows:

1R and 2R: $b_{i,31} = b_{i-1,31}$ $(1 \leq i \leq 31)$;
4R: $b_{i,31} = b_{i-2,31}$ $(48 \leq i \leq 63)$.

In total, there are 46 sufficient conditions.

C.3 Tunnel Technique

We used "Q9 tunnel" in [5], which are based the local collision from step 8 until step 12. The details are shown in Table 4. The crucial idea of Q9 tunnel is that for any message m_8, the chaining variables after the first round will remain the same by modifying only m_9 and m_{12}. m_8, m_9 and m_{12} are used at steps 25, 28 and 32 in the second round, respectively. So the exhaustive search can start from step 25 in the second round. The number of sufficient conditions from step 25 is 23.

Table 4. Tunnel Q9

step index	message	fixed chaining variables
7	m_7	$b_8 = b_7$;
8	m_8	
9	m_9	$b_{10} = 0xffffffff$;
10	m_{10}	$b_{11} = 0x00000000$;
11	m_{11}	

C.4 Collision Attacks on MD4

Sasaki *et al.* [13] published a differential path on MD4 with only 1 sufficient condition located in the third round, which can not be satisfied by the message modification. Here we will only show the sufficient conditions. The message differences are ($\Delta m_0 = 2^{28}$, $\Delta m_2 = 2^{31}$, $\Delta m_4 = 2^{31}$, $\Delta m_8 = 2^{31}$, $\Delta m_{12} = 2^{31}$).

Table 5. Sufficient conditions

Chaining variables	Conditions on bits			
	31 - 24	23 - 16	15 - 8	7 - 0
b_1	1 - - - - - - -	- - - - - - - -	- - - - - - a -	a - - - - - 0 1
b_2	1 - - - - - - -	- - - - 0 - - -	- - - - a - 1 -	0 - - - - - 0 1
b_3	1 - - - - - - 0	- - a a 1 - - -	- - - - 1 - 0 -	0 - - - - - 1 0
b_4	1 - - - - - - 1	a a 1 0 0 - - -	- - - - 0 - 1 a	1 a a a a - - -
b_5	a - - - - - - 0	1 1 0 0 0 a - -	- - - - 0 - 0 1	1 1 1 1 1 - - -
b_6	0 - - - - - - 1	1 1 1 1 0 0 - -	- - - - - - 0 0	0 0 0 0 0 a a -
b_7	0 - - - - - 1 1	0 0 - 0 1 0 - -	- - - - - - 0 1	1 1 1 1 1 1 1 -
b_8	1 - - - - - a 0	0 0 - 1 0 1 - -	- - - 0 - - 0 0	0 0 0 0 1 0 0 -
b_9	0 a a - a a 0 1	0 1 - - - - - -	- - - a - - 1 1	1 1 0 1 1 1 1 -
b_{10}	0 1 1 - 1 0 0 -	1 1 - - - - - -	- - - 1 - - - -	- - - - - - - -
b_{11}	0 0 0 - 1 1 0 -	1 1 - - - - - -	- - - 0 - - - -	- - - - - - - -
b_{12}	0 1 1 a 0 0 1 -	- - - - - - - -	- - - 1 - - - -	- - - - - - - -
b_{13}	- - 1 0 - - 0 -	- - - - - - - -	- - - 0 - - - -	- - - - - - - -
b_{14}	- - 0 0 - - 0 -	- - - - - - - -	- - - 0 - - - -	- - - - - - - -
b_{15}	a - 1 1 - - 1 -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{16}	1 - - a - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{17}	b - - 0 - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{18}	b - - c - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{19}	- - - a - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{20}	a - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{21}	0 - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{22}	c - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{23}	a - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
b_{24}	- - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
...				
b_{33}	0 - - - - - - -	- - - - - - - -	- - - - - - - -	- - - - - - - -
...				

The notation '0' stands for the conditions $b_{i,j} = 0$, the notation '1' stands for the conditions $b_{i,j} = 1$, the notation 'a' stands for the conditions $b_{i,j} = b_{i-1,j}$, 'b' stands for the condition $b_{i,j} \neq b_{i-1,j}$ and 'c' stands for the condition $b_{i,j} = b_{i-2,j}$.