An Algebraic Quality-Time-Advantage-Based Key Establishment Protocol

George T Amariuca
Kansas State University
Manhattan, KS, USA
amariuca@ksu.edu

Sanchita Barman
Iowa State University
Ames, IA, USA
sanchitaB.ISU@gmail.com

Yong Guan
Iowa State University
Ames, IA, USA
guan@iastate.edu

ABSTRACT

The essence of information assurance resides in the ability to establish secret keys between the legitimate communicating parties. Common approaches to key establishment include public-key infrastructure, key-distribution centers, physical-layer security, or key extraction from common randomness. Of these, the latter two are based on specific natural advantages that the legitimate parties hold over their adversaries – most often, such advantages rely on superior or privileged communication channels. This paper tackles a key-establishment protocol that relies on a completely different type of advantage: time. The protocol builds on the idea that when two devices are able to spend a pre-determined, mostly uninterrupted, interval of time in the company of each other, and when such a feat is outside the capability of any realistic attacker, then the legitimate parties should be able to establish a secret key without any prior common information. The paper presents a basic efficient time-based key establishment protocol, and demonstrates how it can be extended to follow customized information transfer functions and deal with predictable fluctuations of wireless interference.

CCS CONCEPTS

• Security and privacy → Key management; Security protocols;

ACM Reference Format:

George T Amariuca, Sanchita Barman, and Yong Guan. 2018. An Algebraic Quality-Time-Advantage-Based Key Establishment Protocol. In *WiSec '18: Proceedings of the 11th ACM Conference on Security Privacy in Wireless and Mobile Networks, June 18–20, 2018, Stockholm, Sweden*. ACM, New York, NY, USA, 10 pages. <https://doi.org/10.1145/3212480.3212507>

1 INTRODUCTION

Information assurance relies on leveraging an advantage that legitimate parties hold over their adversaries. Most often, such advantages are based on pre-shared secret keys. To encrypt and authenticate the information exchanged over the communication channel, the two parties establish session keys, which they derive with the help of either public-key-based protocols (in which public keys are

securely shared with the help of certificates issued by trusted authorities), key-distribution centers (with which each of the parties is assumed to already share a secret key), privileged communication channels (like out-of-band channels [13], or superior wireless channels [3, 19]) or sources of common randomness [1, 4, 12] (like the signal received from a satellite [12] or network metadata [8, 16]).

In this paper, we take a different, and most often overlooked approach. Namely, we aim to design key establishment protocols based solely on the legitimate parties' capability of spending long and uninterrupted (or *quality*) intervals of time in the company of each other – of course, with the assumption that any potential adversary would never be in the position to spend quite the same amount of quality time with either of the two parties. Hence forth, we shall call this the *quality-time advantage*. While our assumption may appear too restrictive for many applications, we argue that it is quite appropriate for many realistic and common ones – and in that respect, it is no less realistic than the assumption of reciprocal or less noisy wireless channels [3, 11, 17–20]. Consider, for example, RFID tags embedded in clothing and grocery packages. They spend most of their time in a secure location, in the company of the home RFID readers. Consequently, they may automatically establish secure communication channels with these readers, over the span of only one night. Similarly, consider a large number of military-grade wireless sensors that are stored in a secure location, for an extended period of time before being deployed in the field (probably by being thrown out of an airplane). They could use their time in the secure location to automatically establish secure channels with each-other, and hence overcome the need for key pre-distribution. In fact it would not even be necessary that they complete the key establishment procedure before deployment – as long as they start it and make sufficient progress in the secure location, they can simply finish the protocol once they successfully land. Another application of the proposed approach is to home-area networks – a newly-acquired device can establish secure channels with the other devices in the home in a completely automatic fashion, over the span of a few days. The same procedure would then be used for periodically re-keying the connection. Of course, to avoid rendering the device unusable for any extended initial period of time, the device could still start working immediately by regular pairing with the other smart home devices – however, the transmission of sensitive information could be blocked until the secure key establishment protocol succeeds.

Finally, while it may appear that the protocol is also applicable to wired networks, great care should be exercised when deploying it in such environments, to ensure that the underlying assumptions of the protocol – namely that potential attackers cannot spend large, uninterrupted periods of time monitoring the legitimate parties

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

WiSec '18, June 18–20, 2018, Stockholm, Sweden

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-5731-9/18/06...\$15.00

<https://doi.org/10.1145/3212480.3212507>

– hold. We believe that such constraints, while easily ensured in certain wireless applications, are generally difficult to enforce in wired networks.

1.1 Motivation

At a first glance, it may seem that the quality-time advantage could be used similarly to a source of common randomness – that is, since the two legitimate parties are in the company of each other for a long, uninterrupted time interval that is not entirely accessible to any adversary, the two parties could simply share common randomness over some communication channel. At the end of the time interval, a secret key would be extracted from this common randomness, with a randomness extractor designed to take into account the adversary's worst-case capability. However, such an approach would suffer from the same drawbacks as the other physical-layer security approaches: namely, the security of the established secret key is *subject to the security of the protocol by which the two legitimate parties decided that they are indeed in a secure environment* (or, for other physical-layer security mechanisms, that they do share a superior channel, or that they can observe a source of common randomness through channels with specified statistics). For our specific case, the two parties would have to be informed of their secure physical location by a human user or another machine, and thus the established secret key would rely on the security of the authentication protocol used to establish the trust in such a user or machine.

The beauty of the quality-time advantage is that it allows us to design a protocol that is completely (1) *automatic* and (2) *independent of other security protocols*, and can hence reside at the root of trust. Note that the first property above completely excludes human interaction (since it is often seen as the weakest link in security protocols), while the second ensures that no prior advantages or shared secret keys can be leveraged – the two legitimate parties trust no-one until the successful completion of our protocol. Interestingly, the second constraint outlined above also excludes the assumption that either of the legitimate parties understands the difference between a physically secure location and an adversary-prone environment.

It should also be mentioned here that while the protocol is indeed time-based, it cannot assume the secure synchronization of the two legitimate parties' time references, nor can it assume that both legitimate parties have reliable time references – in fact, low-cost devices typical of the smart home environment, like RFID tags, have no power source of their own, and hence cannot rely on any absolute or relative notion of time. These devices usually impose additional specific constraints, which include that they might not possess the computational capability (or energy) to engage in sophisticated cryptographic algorithms. It turns out that our proposed protocol is simple enough to be tractable even in the least expensive wireless devices.

1.2 Related Work

Only two main previous works have dealt with time-advantage key establishment. The problem was first introduced by the *adopted-pet protocol* of [2]. The protocol therein is specifically targeted at RFID applications, and is based on breakable ciphers. Namely, [2]

proposes using a simple stream cipher, based on linear-feedback shift registers (LFSR). The RFID tag would clock the stream cipher at each query received from the RFID reader, and would communicate back to the reader the resulting piece of key stream. The (comparatively more sophisticated) RFID reader gathers these *clues* and attempts to find the tag's secret – which is encoded as the LFSR connection polynomial(s). With the right number of clues, the task is rather simple, as it involves solving a linear system of equations. However, if a certain number of clues are missed, the task reduces to solving a system of quadratic equations over a finite field, for which there is no known efficient method (in fact the problem is NP-complete [5] and known as the *MQ problem*). This prevents the adversary from finding the tag's secret. Of course, the legitimate reader may also miss a few clues, and this is solved in [2] by simply stating that a few quadratic terms are tractable by the reader (which can cast them as new unknowns) with access to a few more clues. While the approach of [2] is quite efficient, it has its drawbacks. The LFSR approach leaks information about the tag's secret at a linear rate over time (in this paper, we say that its *information transfer function is linear*) thus failing to achieve one of the goals stated in [2] – namely, that the first derivative of the information transfer function should increase with the length of the time interval that the two legitimate parties spend together, to minimize the amount of information leaked to adversaries. More sophisticated breakable ciphers could be used instead of the LFSR-based ones, but it is not clear how this would be achieved.

Following [2], an alternative solution was patented in [6]. In this patent one of the legitimate devices generates a secret key, and breaks it into several parts (shares). Then it encodes one of the key shares using an erasure code, and broadcasts the resulting codeword over an interval of time. The second legitimate device has to listen for a long enough time interval, in order to be able to recover that share of the key. After obtaining a certain number of shares, the second legitimate device is able to recover the secret key. The main drawback of the approach, however, is that it does not allow the listening device the flexibility to start listening at any time of the broadcast and still recover the key. Namely, the two devices have to be somehow informed of the starting time of the protocol. One could argue that such an event can be triggered by the second device, requesting a new key sharing session. However, if that is the case, then the first device is open to a trivial denial-of-service attack, where the attacker keeps requesting new authentication sessions, which forces the first device to keep generating and sharing new keys until its session-number upper bound is reached, or it becomes hopelessly stuck. Another drawback of [6] is that its information transfer function is not controllable. Due to the use of Shamir's secret-sharing scheme [15], zero information about the secret key leaks until the last needed share is received, after which the entire secret key becomes known. While this feature appears to add security, it may result in a very inflexible protocol, where small changes in the environment may cause complete failure (the protocol does not degrade gracefully).

A related scheme is introduced in [7]. It uses Shamir's secret sharing [15] to share a secret tag key over a large number of RFID tags. The secret key has been distributed in such a way that only the vendor of the associated product would have access to it in the storage room. While our present work bears some similarities with

[7], it has a very different focus and application. In [9], secret sharing is used to prevent random readers from accessing the private information of RFID tags. The *Shamir tags* proposed therein use an algorithm that relies on throttled responses to ensure that the legitimate parties spend with each other a given quantity of time. Finally, our main technique is in essence a puzzle-based approach. While this brings to mind the Merkle puzzles of [14], the current paper does not make use of, or develop algorithms resembling, Merkle puzzles.

1.3 QTAB Key Establishment

By reviewing the related work in the above sub-section, we are now ready to state (and re-state) the main desirable features of our Quality-Time-Advantage-Based (QTAB) key establishment protocol (KEP), and get a first glimpse into what type of techniques are necessary for achieving these features. The QTAB-KEP should: (1) *be completely automatic*, (2) *be independent of other security protocols*, (3) *rely on a single session and be independent of the protocol's starting time*, (4) *allow a customizable information transfer function versus the length of uninterrupted time spent listening to clues*, (5) *be robust to interference causing a few missed clues, or a few erroneous clues*. Since the properties (1) and (2) should already be clear, we shall briefly explain (3), (4) and (5).

The single-session requirement (3) is meant to prevent denial-of-service attacks like those mentioned in Section 1.2 above. The clue-issuing device will maintain a single session, in the sense that it will respond with a single clue, at most once every t_{min} seconds, to any number of queries received in the past t_{min} seconds. Of course, this means that the clue-issuing device maintains a single secret corresponding to the single session.

Now requirement (4) means that, regardless of the specific time instance at which the second legitimate party starts to listen, information about the first party's secret is recovered from the clues, at a rate following a custom function of the length of uninterrupted time spent listening (note that when defining this function – see Figure 3 – time zero is when the second party started to listen to clues, and can correspond to any value of the absolute time). In turn, when considering (3) and (4) together, it should become apparent that the single secret maintained by the first party has to be constantly evolving over time. Otherwise, if the secret remained constant, then two distinct, non-contiguous listening sessions for the same party would result in different information transfer functions, as the first interval already brings information about the key, causing the second interval's information transfer function to start at a higher value than the first interval's information transfer function. The need for (4) arises in the context of graceful degradation, or when some time-based authorization policy is tied to the key establishment protocol. This latter application bears a resemblance to attribute-based encryption – devices that are able to spend longer time intervals with the clue-issuing device get access to more services.

Note that so far we have not dealt with the possibility of the legitimate second party missing some of the clues, or receiving them with errors, even though it does its best to listen for a continuous time interval. Such a phenomenon should be common in wireless

environments. Therefore, we deal with it by imposing feature (5) above. A more formal definition of robustness will be presented in the next section. To achieve (5), our protocol has to use some form of redundancy – in other words, error-correction coding. But since the protocol should be able to run asynchronously, we can only use a rateless code – for this purpose, Shamir's secret sharing scheme [15] (interpreted as a Reed-Solomon error-correction code) appears to be a good candidate. Note that (5) also accounts for some (few) maliciously injected clues. However, should the number of maliciously-injected clues overcome the error-correction properties of the code, the second legitimate party cannot find the secret of the clue-issuing party and thus it is constrained to wait until the two legitimate parties reach the safe environment assumed as part of the problem definition – in a way, the maliciously injected clue problem is solved naturally, without intervention.

2 PROBLEM STATEMENT

With the five protocol features already explained in Section 1.3 above, it is now time to formalize some of them. The following two definitions formulate the notions of robustness and security. It should be noted that these notions implicitly capture multiple of the features (1) through (5) outlined above.

Definition 2.1. Robustness: A QTAB-KEP is said to be (n, k) -robust if a legitimate party who listens to at least n consecutive clues, out of which it can miss at most k clues, can recover the secret key with probability 1.

Definition 2.2. Security: A QTAB-KEP is called (m, p) -secure if an attacker who can listen to at most m consecutive clues, after which she must miss at least p consecutive clues, has no more information about the secret than she had before listening to the first clue.

It is interesting to note that the attacker may listen to multiple blocks of m consecutive clues, as long as any two blocks are separated by blocks of at least p consecutive missed clues. That is, in the worst-case scenario, the attacker has m consecutive recovered clues, followed by p consecutive missed clues, followed by m consecutive recovered clues, followed by p consecutive missed clues, and so on. So, for any integer a , over a span of $am + (a - 1)p$ clues, the attacker has access to only am of them. This definition is motivated by the very assumption underlying our time-based key establishment protocol – namely, that the attacker cannot spend as much uninterrupted time as the legitimate receiver in the company of the clue-issuer – therefore the “breaks” of at least p clues each. Of course, in practice, due to the imperfections of the communication channel (which motivate the first definition above), it is likely that the attacker (similarly to the legitimate receiver) will not be able to acquire all m of the allowed clues – that is, it would probably miss some. Nevertheless, for simplicity, we only consider the worst-case scenario in which all of the allowed m clues are successfully recovered by the attacker.

Going back to Definitions 2.1 and 2.2 above, it may initially appear that they are independent of each other. However, the next two theorems show that this is not the case. In fact, when one chooses the QTAB-KEP parameters (n, k) and (m, p) , the choice of the first tuple severely restricts the choice space of the second.

THEOREM 1. *If there exists a strictly positive integer a such that the following two conditions are satisfied simultaneously:*

$$n - k \leq am, \quad (1)$$

$$k \geq (a - 1)p, \quad (2)$$

then there does not exist a protocol which is (n, k) -robust and (m, p) -secure.

PROOF. There are two possible cases:

Case 1: If $am \leq n - (a - 1)p$, or equivalently $am + (a - 1)p \leq n$, then we conclude that over the span of less than n consecutive clues (since $am + (a - 1)p \leq n$), an attacker should be able to correctly recover am , which is more than $n - k$ (by condition (1)). If the protocol is (n, k) -robust, by Definition 2.1, the attacker should be able to find the secret. Thus, the protocol cannot be secure.

Case 2: If $am > n - (a - 1)p$, or equivalently $am + (a - 1)p > n$, then we conclude that out of a span of more than n consecutive clues (since $am + (a - 1)p > n$), an attacker misses $(a - 1)p$ clues, which is fewer than k clues (by condition (2) above). If the protocol is (n, k) -robust, by Definition 2.1, the attacker should be able to find the secret. Thus, the protocol cannot be secure. $\square$

THEOREM 2. *Assume that there exists a protocol that is (n, k) -robust and for which the following property holds: **(P*)**: **having access to any set of at most $n - k - 1$ clues leaks no information about the secret**. Then, if for all positive integers a conditions (1) and (2) do not hold simultaneously, the protocol is both (n, k) -robust and (m, p) -secure.*

PROOF. We can always find an integer a such that:

$$(a - 1)m < n - k, \quad (3)$$

$$am \geq n - k. \quad (4)$$

Since (4) is the same as (1), by the statement of the theorem we must have that (2) does not hold, so:

$$k < (a - 1)p. \quad (5)$$

Now we have two cases:

Case 1: If $(a - 1)m + (a - 1)p \geq n$, then within any group of n clues, the adversary can only recover at most $(a - 1)m$ clues, which according to (4) is less than $(n - k)$. By property (P*) in the theorem's statement, the adversary has no information about the secret. Hence, the protocol remains (m, p) -secure.

Case 2: If $(a - 1)m + (a - 1)p < n$, then out of any set of n consecutive clues, the adversary has to miss at least $(a - 1)p$, which according to (5) is more than k . Hence the adversary misses too many clues. By property (P*) in the theorem's statement, the adversary has no information about the secret. Hence, the protocol remains (m, p) -secure. $\square$

3 THE BASIC QTAB KEY ESTABLISHMENT PROTOCOL

3.1 Sharing one instance of the secret

Property (P*) in theorem 2 above already hints at a straightforward way of implementing the basic version of the protocol: we shall use threshold cryptography, instantiated by Shamir's secret sharing scheme [15]. For ease of notation, denote the two legitimate

parties by A and B. Now A is the clue-broadcasting device, which broadcasts clues either at regular intervals of time, or upon request (but maintaining a single session at all times – see property (3) in Section 1.3). The adversary is denoted as E. As discussed previously, having to maintain a single session, A will only emit clues at most every t_{min} seconds. Thus, time is discrete – a time instance t_i corresponds to the emission of a clue C_i . In the remainder of the paper, we shall denote vectors by upper-case letters, and scalars by lower-case letters. In this section, single indices will usually correspond to the time instances – for example, s_i is A's secret at time t_i , while double indices will correspond to the pair (time instance, secret) – for example, $s_i = P(c_{i-n+1,i}, c_{i-n+2,i}, \dots, c_{i,i})$ means that A's secret at time i is a function of the clue components of $C_{i-n+1}, \dots, C_i$ corresponding to the secret s_i . In turn, clue C_i emitted at time t_i is a vector of multiple clue components – one for each of the current and next $n - 1$ secrets: $C_i = [c_{i,i}, c_{i,i+1}, \dots, c_{i,i+n-1}]$.

A's secret at time t_i , denoted as s_i , is distributed into n shares, using an (n, k) secret-sharing scheme based on a polynomial code [15]. To recover the secret, B has to gather at least $n - k$ of the shares. The shares are denoted as $c_{i-n+1,i}, c_{i-n+2,i}, \dots, c_{i,i}$ and are transmitted by A at times $i - n + 1, i - n + 2, \dots, i$ respectively.

To implement the protocol, we use a variation of the secret-sharing scheme in [15]. Namely, Shamir's secret sharing works as follows. We choose a large prime number p and settle on the finite field $\mathbb{F} = \mathbb{Z}/p\mathbb{Z}$. We choose and publish n fixed points $(a_1, a_2, \dots, a_n) \in \mathbb{F}$. For every secret s_i , we randomly choose coefficients $f_{1,i}, f_{2,i}, f_{3,i}, \dots, f_{n-k-1,i} \in \mathbb{F}$, and we construct the polynomial $f_i(z) = s_i + f_{1,i}z + f_{2,i}z^2 + f_{3,i}z^3 + \dots + f_{n-k-1,i}z^{n-k-1}$. Note that $f_i(0) = s_i$. Now we compute the shares as $c_{i-n+1,i} = f_i(a_1), c_{i-n+2,i} = f_i(a_2), \dots, c_{i,i} = f_i(a_n)$. Note that any subset of $n - k$ shares completely determines the polynomial, hence making it possible for B to evaluate $f(0)$. To calculate the polynomial in zero, Bob can use Lagrange interpolation for finite fields:

$$f_i(0) = \sum_{j=1}^{n-k} f_i(a_j) \prod_{m=0, m \neq j}^{n-k-1} \frac{a_m}{a_m - a_j}. \quad (6)$$

By contrast, our QTAB secret sharing scheme is based on the systematic Reed-Solomon encoding procedure [10] and works as follows. We work on the finite field $\mathbb{F} = \mathbb{Z}/p\mathbb{Z}$ for a large prime number p . We choose and publish n fixed points $(a_1, a_2, \dots, a_n) \in \mathbb{F}$. We choose $n - k - 1$ random shares $c_{i-n+1,i} = f_{1,i}, c_{i-n+2,i} = f_{2,i}, \dots, c_{i-k-1,i} = f_{n-k-1,i}$ from $\mathbb{F}$. Next, we set $a_0 = 0$ and $f_{0,i} = s_i$, and now, with access to $n - k$ fixed points, we compute the unique polynomial $f_i(z)$ of degree $n - k - 1$, that goes through all these points, i.e. $f_i(a_j) = f_{j,i}$ for $j = 0, 1, 2, \dots, n - k - 1$. This is done using Lagrange interpolation as before:

$$f_i(z) = \sum_{j=0, 1, \dots, n-k-1} f_{j,i} \prod_{\substack{m=0, 1, \dots, n-k-1 \\ m \neq j}} \frac{a_m - z}{a_m - a_j}. \quad (7)$$

Now, the remaining $k + 1$ shares are produced as follows: $c_{i-k+1,i} = f_i(a_{n-k}), \dots, c_{i,i} = f_i(a_n)$. The reason for using the systematic Reed-Solomon construction will be made clear later in this section, when we discuss protocol optimization. As we shall see, it is essential that the protocol works when the first $n - k - 1$ shares are chosen randomly before actually choosing the secret s_i .

To determine the secret when $n - k$ shares become available, B simply computes $f_i(0)$ as in (6) above.

3.2 Computational costs

The systematic Reed-Solomon share generation procedure incurs the following costs at A. The protocol starts by generating $n - k$ random numbers (including the new secret). Then, to compute the polynomial as in (7), A needs to perform $(n - k)^2$ additions and $(n - k)^2$ multiplications – this is calculated under the assumption that the $n - k$ products in (7) are pre-computed and stored as $(n - k)$ -dimensional vectors prior to the beginning of the protocol. Following the computation of the polynomial $f_i(z)$, A has to evaluate it in $k + 1$ points, to produce the remaining $k + 1$ shares. With an efficient algorithm like Horner's, this amounts to $(k + 1)(n - k - 1) \simeq k(n - k)$ additions, and around $k(n - k)$ multiplications. The total cost of running the protocol at A is thus around $n(n - k)$ additions and as many multiplications, i.e. $O(n^2)$.

With pre-computed and stored products in (7), B's cost of running the protocol is merely $n - k$ additions, and as many multiplications, so $O(n)$.

3.3 Error correction capabilities

We already mentioned that the secret-sharing scheme is in essence an error-correction code. The protocol can recover the secret from any subset of $n - k$ shares, but can only correct $\lfloor k/2 \rfloor$ errors (where $\lfloor k/2 \rfloor$ is the largest integer less than or equal to $k/2$). In this sense, a *missed clue* is better than an *erroneously-received* clue: if B receives $n - \lfloor k/2 \rfloor - 1$ shares correctly, and $\lfloor k/2 \rfloor + 1$ in error, then B may end up with the wrong secret, although he has more than $n - k$ correct shares.

We should also note here that the above systematic Reed-Solomon construction does not result in computationally-efficient error-correction decoding [10]. To facilitate error correction, we could use the more efficient BCH representation [10]. In the remainder of this subsection we shall briefly present how the BCH construction can be adapted to our algorithm. Nevertheless, after the current section, we shall continue to refer to the Reed-Solomon representation described above, simply because it is more intuitive, and maps directly to Shamir's secret sharing scheme.

The standard systematic-form BCH implementation of Reed-Solomon codes works as follows. We use the systematic Reed-Solomon procedure illustrated above to produce the polynomial $f_i(z)$ of degree $n - k - 1$. Then, we pick and publish a random primitive element b of the field $\mathbb{F}$, and construct the generator polynomial $g(z) = (z - b)(z - b^2) \dots (z - b^k)$ of degree k . The first $n - k$ shares of the secret are the coefficients of $f(z)$, and the following k shares are the coefficients of the remainder polynomial obtained by dividing $z^k f(z)$ by the generator polynomial $g(z)$. Thus, if $f_i(z) = f_{0,i} + f_{1,i}z + f_{2,i}z^2 + f_{3,i}z^3 + \dots + f_{n-k-1,i}z^{n-k-1}$, and $z^k f_i(z) \bmod g(z) = b_i(z) = b_{0,i} + b_{1,i}z + b_{2,i}z^2 + f_{3,i}z^3 \dots b_{k-1,i}z^{k-1}$, then the shares are simply $f_{0,i}, f_{1,i}, \dots, f_{n-k-1,i}, b_{0,i}, b_{1,i}, \dots, b_{k-1,i}$. Any one of a number of efficient decoding algorithms [10] can be used at this point, and the code is able to correct $\lfloor k/2 \rfloor$ errors.

Two problems appear in this straightforward implementation. First, it is the coefficients of $f_i(z)$, rather than polynomial values of the form $f_i(a_j)$ that constitute the shares. Since the systematic

Reed-Solomon procedure for building $f_i(z)$ assigns the secret s_i to the coefficient $f_{0,i}$, this implies that the entire secret can be recovered from a single coefficient. Second, we need a procedure for constructing $f_i(z)$ that satisfies our requirement that the protocol works when the first $n - k - 1$ shares are chosen randomly before actually choosing the secret s_i . To solve these problems, we present a slightly modified procedure for the construction of $f_i(z)$.

We choose $n - k - 1$ random coefficients $f_{1,i}, \dots, f_{n-k-1,i}$ from $\mathbb{F}$. Next, we pick and publish a random element $a_0 \in \mathbb{F}$ and we assign the secret to be $s_i = f_i(a_0)$. We compute $f_i(z) = f_{n-k-1,i}z^{n-k-1} + \dots + f_{1,i}z + f_{0,i}$, where $f_{0,i} = s_i - (f_{n-k-1,i}a_0^{n-k-1} + \dots + f_{1,i}a_0)$. It should be clear now that $f_i(a_0) = s_i$, and that any subset of $n - k - 1$ coefficients of $f_i(z)$ leaks no information about s_i .

Several important notes should be made here. First, the BCH implementation is appropriate for situations in which the clues are emitted by A (and expected by B) at regular time intervals, but not when the clues are transmitted at random times, or upon request. This is because, in order to correct the errors in the received clues, B has to know the index of each clue – recall that the clues are now the coefficients of polynomials, so their order matters. Alternatively, clues may also contain counters (to help with identifying missing coefficients) in addition to the regular secret shares.

Second, the recovery of the secret from a subset of $n - k$ correctly-received shares (assuming that B is informed by some oracle that these shares are correctly received) can still be done quite efficiently, as long as the indices of the available shares (i.e. the position of the known coefficients in the polynomials $f_i(z)$ and $b_i(z)$) are known. On the other hand, when clues are allowed to be received in error, since the BCH code described above can only correct $\lfloor k/2 \rfloor$ errors, our (n, k) -robust scheme is effectively reduced to an $(n, \lfloor k/2 \rfloor)$ -robust one. That is, in order for B to recover A's secret, B needs to receive at most $\lfloor k/2 \rfloor$ clues out of n in error. Meanwhile the (m, p) -security remains unchanged.

Third, to avoid the problems that arise from erroneously recovered clues, we could simply include in each clue a digest of the secret share, for the purpose of message integrity checking, and ignore any clues that appear to be erroneous – of course, this mechanism would no longer be able to handle maliciously-inserted clues.

3.4 Start-time independence

According to property (3) of the QTAB-KEP, B should be able to start listening to clues at any time throughout A's single session and, if gathering at least $(n - k)$ of the subsequent n clues, should be able to calculate A's secret at some subsequent time instance.

We solve the problem by running n parallel QTAB secret-sharing schemes, each one with a different starting and ending time, like in Table 1. Looking at the table, each column corresponds to a secret, while each row corresponds to the clue transmitted at some time instance. Note that each clue multiplexes n shares – one for each of the current and the following $n - 1$ secrets. That is, the clue at time i is $C_i = [c_{i,i}, c_{i,i+1}, \dots, c_{i,i+n}]$. To recover the secret s_0 at time t_0 , B needs to have at least $n - k$ of the shares transmitted at times $-n + k + 1, \dots, -1, 0$. Looking at Table 1, to recover s_0 B would only use the first column of clues, to recover s_1 B would use the second column of clues, and so forth. More generally, since each clue C_i contains n components, to recover the secret at the current time t_i ,

		•				
		•	•			
		•	•	•		
		•	•	•	•	
		X	•	•	•	
		X	X	•	•	•
		X	X	X	•	•
$t_0 \rightarrow (s_0) :$	X	X	X	X	•	•
$t_1 \rightarrow (s_1) :$	•	X	X	X	X	•
$t_2 \rightarrow (s_2) :$	•	•	X	X	X	X
$t_3 \rightarrow (s_3) :$	•	•	•	X	X	X
$t_4 \rightarrow (s_4) :$	•	•	•	•	X	X
$t_5 \rightarrow (s_5) :$	•	•	•	•	•	X

Table 1: Distribution of shares for the evolving secret in QTAB-KEP

B would use the first component of C_i , the second component of C_{i-1} , the third component of C_{i-2} , and so on.

To implement this multiplexed QTAB scheme, at each new time instance t_i , A starts by generating and transmitting random shares. A transmits $n - k - 1$ such shares, from time t_i to time $t_{i+n-k-1}$. At time $t_{i+n-k-1}$ A chooses a new secret (this would be A's secret s_{i+n} that B can use to authenticate to A at time t_{i+n}), and uses the previously-transmitted $n - k - 1$ shares, along with s_{i+k+1} , to compute the next $k + 1$ shares. He then proceeds to transmit one of the shares at each new time instance, until the last share is transmitted at time $t_i + n$.

3.5 Optimization by batching

Looking at Table 1, we observe that there are two types of shares corresponding to the same secret (i.e., in each column): the ones marked with • and the ones marked with X. This distinction was made to be able to differentiate between the randomly generated $n - k - 1$ shares $c_{i-n+1,i}, c_{i-n+2,i}, \dots, c_{i,i-k-1}$ (denoted by •) and the following $k + 1$ shares $c_{i-k,i}, c_{i-k+1,i}, \dots, c_{i,i}$ (denoted by X) which, given the secret s_i and the first $n - k - 1$ shares (which fix the polynomial), are defined as points of the fixed polynomial.

This distinction is important if we want to batch multiple shares, corresponding to different secrets, into a single share, in order to reduce the length of the clue vector at each time instance. Here is how the batching works. Recall that each secret is shared across $n - k - 1$ •-shares and $k + 1$ X-shares. This results in the clue C_i (a row in Table 1) consisting of $k + 1$ X-shares followed by $n - k - 1$ •-shares. Now, since the •-shares can be chosen randomly, we can make all the •-shares of C_i equal to the last (or right-most) X-share of C_i . This does not affect the security of the protocol.

The result of the batching technique described above is that the clue length is reduced from n to $k + 1$ – for practical protocols, adapted to environments with reasonable interference, the redundancy of the secret sharing scheme is low, relative to the codeword length. This implies that in practice k is a lot less than n , thus resulting in significant savings.

3.6 Parameter selection

So far, we have demonstrated how to design a QTAB-KEP protocol that is (n, k) -robust and satisfies property (P^*) from Theorem 2. Now, by Theorems 1 and 2, the protocol is also (m, p) -secure if and only if for any positive integer a the two conditions (1) and (2) of Theorem 1 do not hold simultaneously. In the remainder of this section, we shall discuss how one might choose the parameter tuple $[(n, k), (m, p)]$ of the QTAB-KEP.

To select the parameters of the QTAB-KEP protocol, we first identify a plausible value for the length of the time interval that the two legitimate parties should be able to spend in the company of each other – say t_n seconds. Then, we decide on the minimum time interval t_{min} seconds between the emission of two clues – this choice has to be made considering plausible energy budgets for both legitimate parties. Then we set $n = t_n / t_{min}$. Next, we have to choose the parameter k , which establishes our coding redundancy. If, for example, we are expecting the environmental (and occasionally malicious) interference to affect 2% of the transmitted clues, then we can pick $k = 5n/100$ to allow room for error correction and a safety margin.

With n and k now fixed, we can look into the choice space of the tuple (m, p) . Recall that the conditions (1) and (2) cannot be satisfied simultaneously for any a . We can easily calculate the complement of the choice space of the tuple (m, p) as follows. For a from 1 to $n - k$, we choose the minimum value of m , and the maximum value of p for which both (1) and (2) are satisfied. That is $m^* = \lceil (n - k)/a \rceil$, and $p^* = \lfloor k/(a - 1) \rfloor$. Then we mark all tuples (m, p) for which $m \geq m^*$ and $p \leq p^*$ as not feasible. What remains is the feasible choice space for (m, p) .

As an example, Figure 1 shows the feasible region (in white, m on the horizontal axis and p on the vertical) when $n = 19$ and $k = 8$, while Figure 2 shows the same for $n = 14$ and $k = 5$. A good choice for the tuple (m, p) would be in one of the right-side corners of the feasible region. However, in practice, a straightforward choice would be $m = n - k - 1$ and $p = k + 1$. To see that this is always a feasible choice, notice that for $a = 1$ we always have $n - k > am$, while for any positive integer $a > 1$ we always have $k < (a - 1)p$ – hence the tuple $(m, p) = (n - k - 1, k + 1)$ is feasible by Theorem 2.

4 CONTROLLING THE INFORMATION TRANSFER

4.1 An extended QTAB-KEP

In the previous section we showed how a robust and secure QTAB-KEP protocol can be achieved using threshold cryptography. The drawback of the protocol, however, is that the information about A's secret remains zero while B gathers the first $n - k - 1$ clues, and jumps to its maximal value at the time of obtaining the $(n - k)$ -th clue. The intrinsic properties of threshold cryptography limit the secret information transfer function to a step function. Thus, in order to achieve a customized information transfer function for our secret, we'll have to approximate it with step functions.

In this section we demonstrate how we can build an enhanced QTAB-KEP that can approximate any monotone increasing information transfer function arbitrarily closely. Consider an arbitrary, monotone increasing function like the one in Figure 3.

We begin by slicing the function into v equal-width horizontal slices, as in Figure 3. We then project the intersection of the slice boundaries with the function onto the x axis, obtaining intervals of different lengths (if the function were linear, we would get equal-length intervals). Denote the lengths of the intervals by $l_1, l_2, \dots, l_v$. Now we have to find integers $z_1, z_2, \dots, z_v$ that maintain the proportions between $l_1, l_2, \dots, l_v$ – that is, $l_1 : l_2 : \dots : l_v = z_1 : z_2 : \dots : z_v$. One way to achieve this is to set $z_j = \lfloor l_j \times 10^d \rfloor$, where the exponent d is chosen so that $z_1 : z_2 : \dots : z_v$ is a good approximation of $l_1 : l_2 : \dots : l_v$.

Now we decompose A 's secret into v sub-blocks of equal length $s_{1,i}, s_{2,i}, \dots, s_{v,i}$, and we run v basic QTAB-KEPs in parallel, one for each secret sub-block. The parameters for the v QTAB-KEPs are denoted as $\{(n_j, k_j), (m_j, p_j)\}, j = 1, 2, \dots, v$ and are chosen such that $(n_1 - k_1) = z_1$, $(n_2 - k_2) = z_1 + z_2$, and so on, until $(n_v - k_v) = z_1 + z_2 + \dots + z_v$. The v QTAB-KEPs are synchronized such that, at any time instance t_i , the latest z_1 clues are enough for finding $s_{1,i}$, the latest $z_1 + z_2$ clues are enough for finding $s_{2,i}$, and so forth, until the latest $z_1 + z_2 + \dots + z_v$ clues are enough for finding $s_{v,i}$. A description of the alignment is given in Table 2. Thus, for example, if B has gathered $z_1 + z_2$ consecutive clues, he should have access to the secret sub-blocks $s_{1,i}$ and $s_{2,i}$, but not to any other sub-block.

It is also important to note that in Table 2, the clues corresponding to secrets $s_{1,i}$ and $s_{2,i}$ correspond to different columns from the two corresponding component QTAB-KEPs. For example, if the clues for $s_{i,2}$ are found on the first column of the second QTAB-KEP (labeled “clues for sub-block s_2 ”), the clues for $s_{i,1}$ would be found in the eighth column of the first QTAB-KEP. This is because the first column of the first QTAB-KEP ends seven time instances before the first column of the second QTAB-KEP. But in order to have the two QTAB-KEPs produce shares of the same secret – that is, $s_{1,i}$ and $s_{2,i}$, rather than $s_{1,i-7}$ and $s_{2,i}$ – the two corresponding columns have to end at the same time t_i .

Figure 1: Feasible region for $n = 19$ and $k = 8$ Figure 2: Feasible region for $n = 14$ and $k = 5$

4.2 Optimization by batching

Similarly to the basic QTAB-KEP of Section 3, the length of a clue $C_{j,i}$ corresponding to time instance t_i and the secret sub-block j (with $i \in \{1, 2, \dots\}$, $j \in \{1, 2, \dots, v\}$) can be restricted to no more than $k_j + 1$ by using the clue-component batching technique of Section 3.5. For the extended QTAB-KEP this would result in a clue length equal to $k_1 + k_2 + \dots + k_v + v$. However, when running the v protocols in parallel, additional batching may be employed across secret sub-blocks. This additional technique can reduce the clue length to $k_1 + k_2 + \dots + k_v + 1$. The savings are thus minimal, and probably not worth the complexity associated with the additional cross-sub-block batching in a real-world implementation.

4.3 Parameter selection

Earlier in this section we proposed to set $(n_1 - k_1) = z_1$, $(n_2 - k_2) = z_1 + z_2$, and so on, until $(n_v - k_v) = z_1 + z_2 + \dots + z_v$. The integers $\{z_j\}$ would be chosen as $z_j = \lfloor l_j \times 10^d \rfloor$, such that $l_1 : l_2 : \dots : l_v \approx z_1 : z_2 : \dots : z_v$. Now, depending on the design constraints (namely, the values of t_n and t_{min} – see Section 3.6), we could choose the exponent d to increase or decrease the

Figure 3: An arbitrary information transfer function

would need to prove not only that it listened for a longer time, but that it also missed proportionally fewer clues – possibly indicating physical proximity.

Alternatively, if larger blocks of missed clues are tolerable as long as the required number of clues are gathered for the secret sub-block that demands the longest time interval, then the error-correction codes associated to the secret sub-blocks may contain increasingly more redundancy, as the number of clues needed for recovering the sub-block increases. Either way, once $\{n_j, k_j\}$ are set, the corresponding values for $\{m_j, p_j\}$ are found in the same manner as before.

4.4 Security considerations

We should note that the Definitions 2.1 and 2.2 were stated for the basic QTAB-KEP, and cannot completely describe the intricacies of the extended QTAB-KEP. At this point, however, it is natural to ask how robust and how secure is the extended QTAB-KEP in terms of these two definitions.

Consider the extended QTAB-KEP described earlier in this section. While it may appear at a first glance that the protocol is (n_v, k_v) -robust, that is not the case. For example, B could receive correctly $(n_1 - k_1 - 1)$ consecutive clues, miss the next k_v consecutive clues, and then receive correctly $(n_v - k_v - n_1 + k_1 + 1)$ more clues at the end. At the end of this time interval, say at time t_i , B should be able to recover the last sub-block of the secret, $s_{v,i}$, but if $k_v > k_1$ (which should hold for most applications) and $(n_v - k_v - n_1 + k_1 + 1) < (n_1 - k_1)$, he will not be able to recover the first sub-block $s_{i,1}$ ¹.

Consequently, for the general case, we can only state a much weaker result, described in Lemma 4.1 below.

LEMMA 4.1. *Let us consider an extended QTAB-KEP consisting of v parallel basic QTAB-KEPs, indexed by $j \in \{1, 2, \dots, v\}$. Assume that the basic QTAB-KEP j is (n_j, k_j) -robust and (m_j, p_j) -secure, and that the parameters (n_j, k_j) were chosen such that $(n_1 - k_1) = z_1$, $(n_2 - k_2) = z_1 + z_2$, and so on, until $(n_v - k_v) = z_1 + z_2 + \dots + z_v$, for some arbitrary positive integers $z_1, z_2, \dots, z_v$. Now denote $n_{\max} = \max\{n_1, n_2, \dots, n_v\}$, $k_{\min} = \min\{k_1, k_2, \dots, k_v\}$, $p_{\max} = \max\{p_1, p_2, \dots, p_v\}$, $m_{\min} = \min\{m_1, m_2, \dots, m_v\}$. Then the extended QTAB-KEP is $(n_{\max}, k_{\min})$ -robust and $(m_{\min}, p_{\max})$ -secure.*

PROOF. From the way in which the values of $(n_j - k_j)$ are chosen, it is clear that $(n_1 - k_1) < (n_2 - k_2) < \dots < (n_v - k_v)$. Now, let us focus on $(n_{\max}, k_{\min})$ -robustness. If B listens over the duration of $n_{\max}$ clues, and misses at most $k_{\min}$ clues, then we could say that $\forall j \in \{1, 2, \dots, v\}$, B listens over the duration of at least n_j clues, out of which he misses at most k_j clues. Since $\forall j \in \{1, 2, \dots, v\}$ the j -th component basic QTAB-KEP is (n_j, k_j) -robust, then B has access

¹The prescribed scenario may seem unnecessarily complicated, but was picked so that the reader is not tricked into believing that the fact that the protocol is not (n_v, k_v) -robust is a mere technicality residing in Definition 2.1. Imagine a scenario in which B could receive correctly $(n_v - k_v)$ consecutive clues, and then miss the next k_v consecutive clues. Clearly, at the end of the protocol – say at time t_i , B can recover $s_{v,i}$, but not $s_{i,1}$. In this case, it appears that the failure of (n_v, k_v) -robustness is only due to a technicality. Namely, for this example B can completely recover all of the secret sub-blocks (and hence the entire secret) at time t_{i-k_v} – it is only at a later time t_i that he cannot recover the first secret sub-block anymore (recall that these sub-blocks change at every new time instance). Thus, we present a more intricate example that shows without a doubt that the protocol is not (n_v, k_v) -robust.

to all the secret sub-blocks $\{s_j\}$. By Definition 2.1, this implies that the extended QTAB-KEP is $(n_{\max}, k_{\min})$ -robust.

The proof of the security is not quite as straightforward, mainly due to the fact that the extended QTAB-KEP, while $(n_{\max}, k_{\min})$ -robust, does not satisfy property (P*) of Theorem 2. Instead, we take advantage of the fact that each of the component basic QTAB-KEPs does satisfy (P*). Because of this, we start by applying Theorem 2 to each individual basic QTAB-KEP. Since the j -th basic QTAB-KEP is (n_j, k_j) -robust and satisfies property (P*), then for any tuple (m, p) , if for all positive integers a we have that either $n_j - k_j > am$ (condition 1 of Theorem 1 is not satisfied) or $k_j < (a-1)p$ (condition 2 of Theorem 1 is not satisfied), then the j -th basic QTAB-KEP is (m, p) -secure. If we can find a tuple (m, p) for which the above holds for any j , then all the component basic QTAB-KEPs are (m, p) -secure, which by definition 2.2 means that no information leaks to the adversary about any of the secret sub-blocks. In turn, this implies that no information leaks to the adversary about the whole secret, which means that the extended QTAB-KEP is (m, p) -secure.

Now all that remains is to show that $\forall j \in \{1, 2, \dots, v\}$, and $\forall a \in \mathbb{Z}_+$, we have that either $(n_j - k_j) > am_{\min}$ or $k_j < (a-1)p_{\max}$. But this is immediately apparent – because each of the component basic QTAB-KEPs is (n_j, k_j) -robust and (m_j, p_j) -secure, by Theorem 1 we have that either $(n_j - k_j) > am_j > am_{\min}$ or $k_j < (a-1)p_j < (a-1)p_{\max}$. $\square$

Let us see what happens when the redundancy of each of the error-correction codes corresponding to the secret sub-blocks is chosen as a percentage of the codeword length (recall that such a choice is the appropriate one if we are solely concerned with combating interference, and we expect such interference to be roughly similar throughout the duration of the protocol – of course, an additional implicit assumption of long enough codewords (or large enough numbers of clues) is made here to ensure that the probability distribution associated with the interference concentrates well enough around its average. In this case, the ordering $(n_1 - k_1) < (n_2 - k_2) < \dots < (n_v - k_v)$ implies the orderings $n_1 < n_2 < \dots < n_v$ and $k_1 < k_2 < \dots < k_v$. If, in addition, we take our own advice from section 3.6 and set $m_j = n_j - k_j - 1$ and $p_j = k_j + 1$, then, by Lemma 4.1 above, the extended QTAB-KEP is (n_v, k_1) -robust and (m_1, p_v) -secure.

Although the above results appear somewhat pessimistic, in practice we can do a bit better at the cost of slightly compromising the notion of robustness. Consider that, if we design a basic QTAB-KEP that is (n_j, k_j) -robust, then we expect that most of the time (say with probability larger than $(1 - \delta)$, for some small δ) a legitimate device engaged in this protocol would be able to correctly receive at least $n_j - k_j$ clues of any set of n_j clues, as long as it is in the company of the clue-issuing device. Otherwise, we should design the QTAB-KEP with larger redundancy. Therefore, with probability $(1 - \delta)^v$, the same device listening over the duration of n_v clues should correctly receive k_v out of these, and also k_{v-1} of the latest n_{v-1} consecutive clues, $\dots$, and also k_1 of the latest n_1 consecutive clues. In this spirit, we introduce the new definition below.

Definition 4.2. δ -Robustness: A QTAB-KEP is said to be (δ, n) -robust if a legitimate party who listens to the clue-issuer for the duration of at least n consecutive clues (out of which some clues may

be received incorrectly), can recover the secret key with probability larger than $1 - \delta$.

With this definition, we can now formulate a probabilistic counterpart to Lemma 4.1 – note that, although robustness fails quite a bit better, the security evaluation needs to remain the same:

LEMMA 4.3. *Let us consider an extended QTAB-KEP consisting of v parallel basic QTAB-KEPs, indexed by $j \in \{1, 2, \dots, v\}$. Assume that the basic QTAB-KEP j is (δ, n_j) -robust and (m_j, p_j) -secure, and that the parameters (n_j, k_j) were chosen such that $(n_1 - k_1) = z_1$, $(n_2 - k_2) = z_1 + z_2$, and so on, until $(n_v - k_v) = z_1 + z_2 + \dots + z_v$, for some arbitrary positive integers $z_1, z_2, \dots, z_v$. Now denote $n_{\max} = \max\{n_1, n_2, \dots, n_v\}$, $p_{\max} = \max\{p_1, p_2, \dots, p_v\}$, $m_{\min} = \min\{m_1, m_2, \dots, m_v\}$. Then the extended QTAB-KEP is $(1 - (1 - \delta)^v, n_{\max})$ -robust and $(m_{\min}, p_{\max})$ -secure.*

PROOF. The proof is quite straightforward. The only modification concerns the definition of robustness, so the security part remains identical to the proof of Lemma 4.1. If B listens over the duration of $n_{\max}$ clues then we could say that $\forall j \in \{1, 2, \dots, v\}$, B listens over the duration of at least n_j clues. Since $\forall j \in \{1, 2, \dots, v\}$ the j -th component basic QTAB-KEP is (δ, n_j) -robust, then B has access to the secret sub-block $s_{i,j}$ with probability larger than $1 - \delta$. Thus, B has access to all of the sub-blocks (so to the entire secret s_j) with probability at least $(1 - \delta)^v$. By Definition 4.2, this implies that the extended QTAB-KEP is $(1 - (1 - \delta)^v, n_{\max})$ -robust. $\square$

The advantage of the new Lemma 4.3 is that one can now design the extended QTAB-KEP at much smaller redundancy levels – instead of trying to make sure that the value of $k_{\min}$ provides enough redundancy for a codeword of length $n_{\max}$, we can now simply design each of the component basic QTAB-KEPs individually, thus being able to achieve smaller overall values for k_j , and consequently better overall values for $m_{\min}$ and $p_{\max}$.

5 CONCLUDING REMARKS

In this paper, we have introduced two protocols for key establishment between legitimate parties, based solely on quality time advantages (QTAB-KEPs). The first is the basic QTAB-KEP, and is based on a slight modification of the threshold cryptography technique of [15]. While useful in practice, the basic protocol lacks the ability to control its own information transfer function, making it inappropriate when graceful performance degradation or time-based authorization policies are a requirement. The second protocol is the extended QTAB-KEP, which uses multiple basic QTAB-KEPs with different parameters, aligned in parallel, and features completely customizable information transfer functions, as well as customizable redundancy levels over the duration of the protocol. We formalized the notions of robustness and security for the basic QTAB-KEP, and evaluated both the basic and the extended protocols in the context of these definitions. While the extended QTAB-KEP can be proven to be $(n_{\max}, k_{\min})$ -robust and $(m_{\min}, p_{\max})$ -secure, we found this characterization to be quite pessimistic. While the security description cannot be improved upon, we found that for practical purposes using a new probabilistic definition – namely that of δ -robustness – can lead to significant design improvements. Future work may concentrate on mechanisms for further reducing

clue size, and on alternative implementations of the QTAB-KEP, providing more relaxed forms of security (perhaps transitioning from the information-theoretic, perfect security of threshold cryptography to security against computationally-bounded adversaries).

6 ACKNOWLEDGEMENT

This work was partially supported by NIST CSAFE under Cooperative Agreement No. 70NANB15H176, NSF under grants No. CNS-1527579, CNS-1619201, and CNS-1730275, and Boeing Company. We appreciate anonymous reviewers for their valuable suggestions and comments.

REFERENCES

- [1] R. Ahlswede and I. Csiszar. 1993. Common Randomness in Information Theory and Cryptography – Part I: Secret Sharing. *IEEE Trans. Inform. Theory* 39 (July 1993), 1121–1132.
- [2] George T. Amariuca, Clifford Bergman, and Yong Guan. 2011. An Automatic, Time-Based, Secure Pairing Protocol for Passive RFID. In *RFID. Security and Privacy - 7th International Workshop, RFIDSec 2011, Amherst, USA, June 26-28, 2011, Revised Selected Papers*. 108–126. https://doi.org/10.1007/978-3-642-25286-0_8
- [3] I. Csiszar and J. Körner. 1978. Broadcast Channels with Confidential Messages. *IEEE Trans. Inform. Theory* 24 (May 1978), 339–348.
- [4] I. Csiszar and P. Narayan. 2008. Secrecy Capacities for Multiterminal Channel Models. *IEEE Trans. Inform. Theory* 54, 6 (June 2008), 2437–2452.
- [5] A.S. Fraenkel and Y. Yesha. 1979. Complexity of problems in games, graphs and algebraic equations. *Discrete Applied Mathematics* 1, 1 (1979), 15 – 30. [https://doi.org/10.1016/0166-218X\(79\)90012-X](https://doi.org/10.1016/0166-218X(79)90012-X)
- [6] A. Juels. 2014. Secret sharing in cryptographic devices via controlled release of plaintext information. (July 8 2014). <https://www.google.com/patents/US8774410> US Patent 8,774,410.
- [7] Ari Juels, Ravikanth Pappu, and Bryan Parno. 2008. Unidirectional Key Distribution Across Time and Space with Applications to RFID Security. In *Proceedings of the 17th Conference on Security Symposium (SS'08)*. USENIX Association, Berkeley, CA, USA, 75–90. <http://dl.acm.org/citation.cfm?id=1496711.1496717>
- [8] Mohammad Reza Khalili-Shoja, George Traian Amariuca, Shuangqing Wei, and Jing Deng. 2016. Secret Common Randomness From Routing Metadata in Ad Hoc Networks. *IEEE Transactions on Information Forensics and Security* 11, 8 (2016), 1674–1684.
- [9] Marc Langheinrich and Remo Marti. 2007. Practical Minimalist Cryptography for RFID Privacy. *IEEE Systems Journal* 1, 2 (2007), 115–128. <https://doi.org/10.1109/JSYST.2007.907683>
- [10] Shu Lin and Daniel J Costello. 2004. *Error control coding*. Vol. 2. Prentice Hall Englewood Cliffs.
- [11] Suhas Mathur, Narayan M, Chunxuan Ye, and Alex Reznik. 2008. Radio-telepathy: extracting a secret key from an unauthenticated wireless channel. In *In MobiCom '08*. 128–139.
- [12] U. E. Maurer. 1993. Secret Key Agreement by Public Discussion from Common Information. *IEEE Trans. Inform. Theory* 39 (May. 1993), 733–742.
- [13] Rene Mayrhofer and Hans Gellersen. 2007. Shake well before use: authentication based on accelerometer data. *Pervasive Computing* 4480 (2007), 144–161.
- [14] Ralph C. Merkle. 1978. Secure Communications over Insecure Channels. *Commun. ACM* 21, 4 (April 1978), 294–299. <https://doi.org/10.1145/359460.359473>
- [15] Adi Shamir. 1979. How to Share a Secret. *Commun. ACM* 22, 11 (Nov. 1979), 612–613. <https://doi.org/10.1145/359168.359176>
- [16] Mohammad Reza Khalili-Shoja, George Traian Amariuca, Zhengdao Wang, Shuangqing Wei, and Jing Deng. 2017. Asymptotic converse bound for secret key capacity in hidden Markov model. In *Information Theory (ISIT), 2017 IEEE International Symposium on*. IEEE, 1968–1972.
- [17] J.W. Wallace and R.K. Sharma. 2010. Automatic Secret Keys From Reciprocal MIMO Wireless Channels: Measurement and Analysis. *Information Forensics and Security, IEEE Transactions on* 5, 3 (Sept. 2010), 381–392. <https://doi.org/10.1109/TIFS.2010.2052253>
- [18] R. Wilson, D. Tse, and R.A. Scholtz. 2007. Channel Identification: Secret Sharing Using Reciprocity in Ultrawideband Channels. *Information Forensics and Security, IEEE Transactions on* 2, 3 (2007), 364–375. <https://doi.org/10.1109/TIFS.2007.902666>
- [19] A. D. Wyner. 1975. The Wire-Tap Channel. *The Bell System Technical Journal* 54 (Oct. 1975), 1355–1387.
- [20] Chunxuan Ye, S. Mathur, A. Reznik, Y. Shah, W. Trappe, and N.B. Mandayam. 2010. Information-Theoretically Secret Key Generation for Fading Wireless Channels. *Information Forensics and Security, IEEE Transactions on* 5, 2 (June 2010), 240–254. <https://doi.org/10.1109/TIFS.2010.2043187>