Simple and Communication Complexity Efficient Almost Secure and Perfectly Secure Message Transmission Schemes

Yvo Desmedt[1,2,⋆], Stelios Erotokritou[1,⋆⋆], and Reihaneh Safavi-Naini[3,⋆⋆⋆]

[1] Department of Computer Science, University College London, UK
{y.desmedt,s.erotokritou}@cs.ucl.ac.uk
[2] Research Center for Information Security (RCIS), AIST, Japan
[3] Department of Computer Science, University of Calgary, Canada
rei@ucalgary.ca

Abstract. Recently Kurosawa and Suzuki considered almost secure (1-phase n-channel) message transmission when $n = (2t + 1)$. The authors gave a lower bound on the communication complexity and presented an exponential time algorithm achieving this bound. In this paper we present a polynomial time protocol achieving the same security properties for the same network conditions.

Additionally, we introduce and formalize new security parameters to message transmission protocols which we feel are missing and necessary in the literature.

We also focus on 2-phase protocols. We present a protocol achieving perfectly secure message transmission of a single message with $O(n^2)$ communication complexity in polynomial time. This is an improvement on previous protocols which achieve perfectly secure message transmission of a single message with a communication complexity of $O(n^3)$.

Keywords: Information-theoretic cryptography, Perfectly secure message transmission, Almost secure message transmission, Cryptographic protocols, Privacy, Reliability, Coding theory, Secret sharing.

1 Introduction

Perfectly secure message transmission (PSMT) schemes were introduced by Dolev et al. in [2]. Such protocols aim to provide perfect privacy and perfect reliability of message transmission between two parties (a sender and a receiver) who do not share a key, in the presence and influence of an infinitely powerful

[⋆] This work was done while funded by EPSRC EP/C538285/1 and by BT, as BT Chair of Information Security.

[⋆⋆] The author would like to thank EPSRC and BT for their funding of this research.

[⋆⋆⋆] This research was supported in part by iCORE (Informatics Circle of Research Excellence) in the Province of Alberta, as well as NSERC (Natural Sciences and Engineering Research Council) in Canada.

active adversary. These protocols are important as they are fundamental cryptographic protocols with important applications in research [3,5]. In the (r-phase, n-channel) model, a sender wishes to securely send a secret to a receiver. This transmission should occur in r phases with n channels connecting the sender and receiver. The adversary is able to corrupt up to t network nodes.

In [2] it was shown that if $r = 2$, $n \geq (2t + 1)$ is a necessary and sufficient condition for PSMT. In this paper we present a protocol which is the most efficient in the literature for the transmission of a single message. Our protocol achieves this in polynomial time and with $O(n^2)$ communication complexity. This improves on the $O(n^3)$ communication complexity of previous protocols. Optimum results for two phase protocols with regards to transmission rate have only recently been achieved. One of these results was that of [1] at CRYPTO 2006. The paper presented a protocol with $O(n)$ transmission rate but exponential computational complexity. These results were improved at EUROCRYPT 2008 in [7] where the authors presented a polynomial 2-phase PSMT protocol able to achieve $O(n)$ transmission rate. The communication complexity for this protocol is however $O(n^3)$, while the one we present is $O(n^2)$. Despite the work of [7] which achieves linear transmission rate the protocol they present is only optimal for communication with a large number of messages. For applications which require the perfectly secure transmission of a single message, the protocol we present is the most efficient in the literature.

Different variations on the security of message transmission schemes exist in the literature [2,6,9,13]. In [2] the authors showed that a necessary and sufficient condition for PSMT to occur when $r = 1$ is that $n \geq 3t+1$. For these conditions the authors also provided a protocol achieving perfect security with optimum communication complexity. In [4] the authors considered perfect security with probabilistic reliability. In such protocols, reliability of message transmission may fail with a bounded probability. The work was important as it presented the sizable gap between the connectivity required to achieve perfect as opposed to probabilistic security. In [8] Kurosawa and Suzuki considered almost secure (1-phase, n-channel) message transmission when $n = 2t+1$. The authors showed that almost secure message transmission can be achieved provided we are willing to accept a small probability that the protocol will fail. A lower bound in communication complexity for this type of transmission was given and an exponential time protocol achieving this bound was presented. Srinathan et al. [12] improved on this by presenting a polynomial time protocol achieving similar security. We also present a polynomial time almost secure message transmission protocol. Contrary to the protocol presented in [12] which can transmit $O(n)$ messages, the protocol we present is for the transmission of a single message only. The protocol we present is computationally more efficient requiring in the order of n less computation than the protocol of [12]. For applications where the almost secure transmission of a single message is required, the protocol we present is the most efficient and appropriate to use.

Both protocols we present are of interest to both theoretical and practical purposes. Whereas previous protocols have sought to optimize transmission rate only [7,8,12], we have concentrated on making protocols more efficient with regards to communication and computation complexities. The protocols we present are theoretically the most efficient regarding these complexities. They are also important for practical purposes. In real world practical applications, computational cryptography is generally used. When these are not available or are broken, in order to initialize these, new keys will have to be exchanged between communicating parties. In such circumstances it makes no sense to send multiple messages.

In this paper we also look at different security definitions for message transmission schemes which exist in the literature. These seem to vary from paper to paper and different aspects to security of message transmission are termed in a different way by various authors. We aim to resolve this confusion by introducing and formalizing new security parameters to message transmission protocols which we feel are missing and necessary in the literature.

2 Background

2.1 Environment of Message Transmission Protocols

In a message transmission protocol the sender starts with a message M^A drawn from a message space $\mathcal{M}$ with respect to a certain probability distribution. We assume that the message space $\mathcal{M}$ is a subset of a finite field $\mathbb{F}$. At the end of the protocol, the receiver outputs a message M^B. We consider a synchronous setting in which messages between sender and receiver are sent in phases. A phase is thus a transmission from sender to receiver or vice-versa.

We will assume an unconditionally secure setting where the adversary is computationally unbounded. We consider an active adversary which can take control of t internal nodes in a network. The adversary is assumed to know the complete protocol specification, message space $\mathcal{M}$ and the complete structure of the network graph. We will only consider static adversaries which must choose the nodes to corrupt before the protocol begins. The adversary is able to view all the behavior (randomness, computation, messages received) and can take full control of any node which is compromised.

The communication network is modeled as a directed graph $G = G(V, E)$ whose nodes are the parties and whose edges are point-to-point reliable and private communication channels. The network model which will be considered consists of bidirectional channels between a sender and a receiver. Disjoint paths between a sender and a receiver are referred to as wires and will be termed so from now on. As the adversary can take control of t internal nodes in a network, the adversary has the capability to control t wires.

2.2 Security Definition

Here we present the current security definitions for message transmission protocols which exist in the literature. We argue why we think these definitions are incomplete and formalize new security parameters we believe should be included.

Current Security Definition - Probabilistic Reliability. The following definition is taken from Franklin and Wright [4] and defines (ϵ, δ)-security.

1. Let $\delta < \frac{1}{2}$. A message transmission protocol is δ-reliable if, with probability at least 1 - δ, B terminates with $M^B = M^A$. The probability is over the choices of M^A and the coin flips of all nodes.
2. A message transmission protocol is perfectly reliable if it is 0-reliable.
3. ϵ refers to the privacy that is achieved. As we will be considering perfect privacy we refer the reader to [4] for the definition of ϵ-privacy.
4. A message transmission protocol is (ϵ, δ)-secure if it is ϵ-private and δ-reliable.

Current Security Definition - Probabilistic Failure. Almost secure message transmission was considered in [8] with the following security definition:

Definition 1. *We say that a (1-phase, n-channel) message transmission scheme is (t, δ)-secure if the following conditions are satisfied for any adversary* ***A*** *who can corrupt at most t out of n channels.*

Privacy. ***A*** *learns no information on M^A. More precisely when R denotes the random variable,*

$$Pr(R = M^A | X_{i_1} = x_{i_1}, \cdots, X_{i_t} = x_{i_t}) = Pr(R = M^A)$$

for any $M^A \in \mathcal{M}$ and any possible $x_{i_1}, \cdots, x_{i_t}$ messages observed by ***A*** *on adversary controlled wires.*

General Reliability. *The receiver outputs $M^B = M^A$ or $\perp$ (failure). The receiver thus never outputs a wrong secret.*
Failure. *$Pr($Receiver outputs $\perp) < \delta$.*

Comparison of the Two Security Definitions. The above security definitions refer to two very different security properties. The definition of reliability from [4] only considers executions that terminate and always output a message. The protocol outputs the correct message with a bounded probability. However, the reliability definition of [8] identifies two types of executions. The first of these are executions that terminate but do not produce an output (i.e. output $\perp$). The second of these are executions that terminate and always produce the correct result. Reliability in these definitions refers to both types of executions.

We propose a new security definition that reconciles the above definitions. This is done by introducing new security parameters which capture the *authenticity* of the received message and the *availability* of the message transmission protocol. Executions that terminate but do not produce a correct output can be seen as an attack on authenticity of the message. Executions that output failure can be seen as an attack on the availability of the transmission protocol. This leads us to propose the following definition.

New Security Definition. The first security parameter we call the availability of a transmission protocol and is defined as follows:

– Let $\gamma \leq 1$. A message transmission protocol achieves γ-availability if, with probability at least 1 - γ, B accepts a message, i.e. B accepts $\perp$ with probability γ.

Availability considers executions that were captured by the δ parameter in the security definition of [8].

Authenticity is the second security parameter we introduce. Assuming the sender transmits M^A and the receiver accepts $M^B \in \{\mathcal{M}, \perp\}$, δ-authenticity is achieved by the following conditional probability.

$$\delta = P(M^A \neq M^B | ReceiverAccepts).$$

The above covers both the reliability definition of [4] and the executions of [8] which terminate and always produce the correct result.

ϵ-privacy remains the same as defined in [4]. This type of security from now on will be referred to as $(\epsilon, \delta, \gamma)$-security. It should be pointed out that perfectly secure message transmission protocols (i.e. the work of [2,6,10]) achieve $(0, 0, 0)$-security under this new definition. In our work we will present a polynomial 1-phase $(0, 0, \gamma)$-secure protocol in Section 3 and in Section 5 we consider 2-phase $(0, 0, 0)$-secure protocols.

2.3 Secret Sharing Schemes

An m-out-of-n threshold secret sharing scheme allows for a secret message M^A to be distributed as a selection of n shares $\{s_1,\ldots,s_n\}$ so that the following properties are achieved:

1. Any collection of m shares is able to reconstruct the secret message M^A.
2. Any subset of $(m - 1)$ or less shares reveals no information about M^A.

Variant of Shamir Secret Sharing Scheme. Shamir secret sharing [11] allows for the reconstruction of a secret using polynomial interpolation. When secret sharing is used in the literature it is assumed that the x-coordinates of points to be used are available in a public database and shares sent to participants (or in our case across wires) are the y-coordinates of the points. When using Shamir secret sharing in message transmission protocols, denoting as p the polynomial from which the shares are constructed, it is usually the norm to transmit share $p(i)$ across wire w_i - where $1 \leq i \leq n$ and n denotes the number of wires connecting sender and receiver.

In this paper we will use a variant of the above for the protocol of Section 3. The only thing that will vary is that the x-coordinates of points to be used will be private and shares sent will be the (x, y)-coordinates of the points.

3 Polynomial 1-Phase Almost Secure Message Transmission

We now describe our 1-phase almost secure message transmission protocol for $n = (2t + 1)$. As our protocol is relatively simple we first present the main idea of our protocol. We then describe the main techniques used in the protocol. We then formally present the security and complexity proof - showing that it is a polynomial algorithm regarding computation and communication complexities.

3.1 Main Idea

As our protocol is a 1-phase protocol, communication can only occur one way from sender to receiver. Denoting as M^A the secret message of the transmission, the sender will construct a $(t+1)$-out-of-n secret sharing code of M^A. The sender thus has n shares $(s_1, \ldots, s_n)$ of M^A. For each one of the n shares the sender will then carry out a $(t+1)$-out-of-n secret sharing (these new shares are termed as sub-shares for clarity) and will transmit these sub-shares to the receiver - the way this is done is outlined in Section 3.2. The receiver will then check the correctness of the shares of M^A and considering only the correct shares will proceed to carry out error detection. This is also outlined in Section 3.2. If no error is detected the receiver accepts the message interpolated by the shares. If at least one error is detected, the receiver accepts $\perp$. When the receiver accepts a value the protocol terminates. As in some cases the receiver accepts $\perp$ and in all other cases the receiver accepts the correct message with perfect secrecy and authenticity the protocol achieves $(0, 0, \gamma)$-security.

3.2 Main Protocol Techniques

In this section we outline three main techniques used in the protocol. The first is the encoding and transmission of the secret message executed by the sender. The other two are carried out by the receiver and are the identification of faulty wires and error detection schemes.

Message Encoding and Transmission. Denoting as M^A the secret message of the transmission, the sender will carry out a $(t+1)$-out-of-n secret sharing of M^A - obtaining shares $(s_1, \ldots, s_n)$. The sender does this by choosing a random polynomial p of degree at most t over $GF(q)$ such that $p(x) = M^A + a_1x^1 + \cdots + a_tx^t$ - where $a_1, \ldots, a_t$ are uniformly random elements of $GF(q)$ and $q \gg n$ denotes the size of the finite field. The n shares $(s_1, \ldots, s_n)$ are obtained by evaluating $(p(1), \ldots, p(n))$.

For $1 \leq i, j \leq n$ the sender proceeds to construct a $(t+1)$-out-of-n secret sharing scheme of share s_i and transmit the constructed shares in the following manner:

1. The sender chooses a random polynomial p_i of degree at most t over $GF(q)$ such that $p_i(x) = s_i + a_{i1}x^1 + \cdots + a_{it}x^t$ where $a_{i1}, \ldots, a_{it}$ are uniformly random elements of $GF(q)$.
2. n different uniformly random elements $(r_{i1}, \ldots, r_{in})$ over the finite field are then selected.
3. The n sub-shares of s_i are computed by evaluating p_i at $(r_{i1}, \ldots, r_{in})$ to obtain $(s_{i1} = p_i(r_{i1}), \ldots, s_{in} = p_i(r_{in}))$.
4. The random elements with the corresponding sub-shares are coupled together to obtain $((r_{i1}, s_{i1}), \ldots, (r_{in}, s_{in}))$.

5. The definition of polynomial p_i is transmitted over wire w_i.
6. The sender transmits the pair (r_{ij}, s_{ij}) over wire w_j.

Faulty Wire Detection. This technique is carried out by the receiver at the end of the protocol phase. Identification of faulty wires is done in the following way:

```
– Initialize set $FAULTY := \emptyset$, $REPEAT_{FLAG} := TRUE$.
– Do the following while ($REPEAT_{FLAG} = TRUE$):
  a $REPEAT_{FLAG} := FALSE$.
  b For $i := 1, \ldots, n$
     1. IF wire $w_i \in FAULTY$ GOTO Step 6.
     2. Denote as $p_i$ the polynomial definition received from wire $w_i$.
     3. Considering only the $i^{th}$ pair of values - $(r_{ji}, s_{ji})$, received from wires
        $w_j \notin FAULTY$.
     4. IF $p_i(r_{ji}) = s_{ji}$ for at least $(t+1)$ pair of values received from
        different wires, then do nothing.
     5. ELSE wire $w_i$ is identified as a faulty wire. Add $w_i$ to $FAULTY$.
        Set $REPEAT_{FLAG} := TRUE$.
     6. End of loop.
```

The faulty wire detection scheme described above identifies faulty wires. However, it cannot guarantee that wires identified by the scheme as non-faulty are not controlled by the adversary and that changes were not carried out. As will be outlined later, adversary controlled wires can still pass the test of this scheme - even if changes were carried out. In short, adversary controlled wires can achieve this if any alterations carried out on polynomial definitions still result in the algorithm finding at least $(t+1)$ pair of values passing the test of Step 4.

It is easy to see that the computational complexity of the above scheme is polynomial. The while loop can be repeated at most t times (as overall at most t faulty wires can be identified in different instances of the while loop).

Error Detection. Error detection is carried out in the following manner. Considering only wires $w_i \notin FAULTY$, evaluate $p_i(0)$ for the polynomial received from wire w_i to obtain share s_i and denote as m the number of shares obtained. Carry out error detection on the m shares in the following manner. Select $t+1$ random shares from the m shares to obtain polynomial p. If any of the remaining $m-(t+1)$ shares do not lie on p then an error has been detected and the receiver outputs $\perp$. Otherwise the receiver accepts $p(0)$ of the obtained polynomial as the message of the transmission.

It is clear to see that the computational requirements of this error detection process is polynomial.

3.3 Security and Efficiency

Theorem 1. *The above protocol achieves* $(0, 0, \gamma)$ *security for appropriately large* q.

Proof. We first prove the perfect privacy of the protocol. The secret message M^A is secret shared using a $(t+1)$-out-of-n secret sharing scheme. The adversary can only learn t of these shares - those whose polynomial definitions are sent on adversary controlled wires. For the secret sharing of the remaining shares the adversary only learns t sub-shares of each and thus cannot learn any of these shares. As the adversary learns t shares, no information is obtained about the secret message. The protocol thus achieves perfect privacy.

Perfect authenticity is achieved as the receiver accepts a message only when no errors are detected. This is proven by the following lemma.

Lemma 1. *The faulty wire detection scheme will always identify as correct wires the set of* $t+1$ *non-faulty wires.*

Proof. The faulty wire detection scheme identifies as a correct wire those whose polynomial definitions correspond to at least $(t+1)$ received sub-shares. As honest wires do not alter any information and as there are at least $(t+1)$ honest wires, honest wires will always be identified as correct wires. □

Due to the above lemma, at least $(t+1)$ original shares of M^A will be considered by the sender in the error detection scheme. Because of this, no matter what changes the adversary carries out the receiver will never accept a wrong message. This is because the degree of the polynomial used in the secret sharing of M^A is at most t. This means that any alterations the adversary carries out cannot result in a different t-degree polynomial which includes all the honest $(t+1)$ shares (corresponding to honest wires) of M^A . Therefore no matter what alterations the adversary carries out to share values, at least one error will always be detected. The receiver will thus never accept a message different to the message sent by the sender. Perfect authenticity is therefore achieved.

We now calculate the availability. Failure of message transmission (i.e. when the receiver outputs $\perp$) occurs when an error is detected in the error detection scheme of Section 3.2. An error is detected by this technique only when the adversary successfully alters at least one share of the secret message. This can only be achieved when a polynomial definition transmitted over a faulty wire is altered and after the execution of the faulty wire detection technique, the specific wire does not belong in the set $FAULTY$. For this to occur the adversary must ensure that at least $(t+1)$ sub-shares received over all wires lie on the altered polynomial. Assuming that the adversary controls t wires (all of which do not belong in the set $FAULTY$) - and in turn controls t sub-shares, any strategy followed by the adversary must ensure that *at least one sub-share* not controlled by the adversary lies on the polynomial definition to be altered by the adversary.

The reader is reminded that shares in the scheme are a pair (r_x, s_y) where r_x is a random field element representing the x-value of a point on a 2-dimensional plane and s_y is the evaluation of r_x for a particular polynomial. As the adversary knows the polynomial definition transmitted on adversary controlled wires, the adversary knows the value of all possible sub-shares ((r_x, s_y) pairs) that could be constructed using that polynomial. What the adversary does not know is which specific $(t+1)$ sub-shares were transmitted over the honest wires. As the values

r_x were uniformly randomly chosen by the sender, in order for the adversary to succeed in an attack against the protocol availability, the best strategy the adversary could follow is to guess these r_x values. (Note that from the original polynomial and a correct r_x value, the adversary knows the corresponding s_y.) Given that the adversary controls t wires the adversary is able to rule out t of these r_x possible values over the finite field - leaving $(q - t)$ possible values for r_x sent over the $(t + 1)$ honest wires.

As mentioned earlier, it is sufficient that *at least one sub-share* not controlled by the adversary lies on the altered polynomial. The adversary can try to guess up to t of these sub-shares[1]. The probability that at least one is successful, is one minus the probability that all t guesses are wrong. All being wrong means, that all t were not chosen by the sender, which obviously means they came from the $q - (2t + 1)$ remaining ones. So, the probability of success is then given by:

$$1 - \frac{\binom{q-2t-1}{t}}{\binom{q-t}{t}}$$

The above analysis is for the case the adversary decides to alter a single polynomial. If the adversary were to change the polynomial definition of two wires and the attack of one wire failed, this would mean that for the other wire attack to succeed, it would require for two sub-shares transmitted over honest wires to be guessed correctly. This makes the full analysis more complex and due to space reasons will appear in the full version of the text. □

We now analyze the complexity of the protocol. Denoting as $|\mathbb{F}|$ the bit length of the field elements, the communication complexity of the protocol is $O(n^2|\mathbb{F}|)$. The computational complexities of both sender and receiver are polynomial.

We have thus presented a polynomial almost secure polynomial protocol improving on the exponential time protocol presented in [8].

4 Comparison of Protocol to Previous Work

In this section we compare the protocol presented in the previous section to previous work and argue as to why it is a valuable addition to the knowledge.

Comparison of Protocol to Suzuki and Kurusawa Protocol. The work presented by Suzuki and Kurusawa in ICITS 2007 [8] was important as it proved the lower bound of communication complexity required for almost secure message transmission. Despite the authors presenting a protocol achieving this bound,

[1] As the degree of the polynomial is at most t the adversary cannot guess more that t of the remaining sub-shares. This is because $(t + 1)$ correct sub-shares interpolate the original polynomial. If the adversary guesses more than t sub-shares and more than t are correct this means that the original polynomial will be interpolated and in effect the adversary carries out no changes.

the computational complexity of the protocol was exponential which makes the protocol inefficient with regard to time for large values of n.

The protocol presented in the previous section, despite having a greater communication complexity of $O(n^2)$ - as opposed to the optimal $O(n)$, is of polynomial computation time making it a more appropriate protocol for use.

Comparison of Protocol to Srinathan et al. Protocol. The work presented by Srinathan et al. in PODC 2008 [12] was important as it presented an almost secure message polynomial time transmission protocol which achieves the optimal transmission rate of $O(n)$ with a communication complexity of $O(n^2)$. For the protocol to achieve this transmission rate $O(n)$ messages are transmitted.

The protocol presented in the previous section also has a communication complexity of $O(n^2)$, but only a single message is transmitted.

When these two protocols are compared it may seem that the Srinathan et al. protocol is the best protocol to use as more messages can be sent with the same communication complexity.

Even though this is true, if the two protocols were to be compared with respect to their computational complexities, it will be found that the protocol presented in the previous section requires much less computation to complete than the Srinathan et al. protocol.

The protocol of Srinathan et al. decides on whether to accept a message or $\perp$ using error detection. For the Srinathan et al. protocol to accept a message a total of n error detections need to be carried out. For $\perp$ to be accepted between one and n error detections need to be carried out - the actual number depending on the actions of the adversary. Contrary to this, the protocol presented in the previous section accepts a message or $\perp$ within only a single error detection. Because of this, the protocol presented in this paper requires in the order of n less computation than the protocol of Srinathan et al.

This makes the presented protocol very useful for situations where the almost secure transmission of a single message is required. This can include the transmission of an encryption key in wireless sensor networks. In such a situation the keys can be transmitted with the least amount of computation carried out by receivers - in this case wireless sensors, which is an important factor for the preservation of the wireless sensor battery life.

Note: Transforming the Desmedt-Wang Protocol to a $(0, 0, \gamma)$ Protocol. We now describe how to transform the protocol presented in Section 3 of [15] as a $(0, \delta)$ protocol (following the security definition of [4]) to a $(0, 0, \gamma)$ protocol with $\gamma = \delta$.

The protocol works by sharing a secret using a $(t + 1)$-out-of-$(2t + 1)$ secret sharing scheme. Using message authentication codes (MAC's) each share is authenticated $(2t + 1)$ different times using authentication keys specific to each wire. Each share is then sent to the receiver only once and upon each wire only one share is sent. The authentication codes of the shares are sent over the same wire the share is sent on and the authentication keys are sent on their respective wires. Although not stated in the description of the protocol, this process can be carried out in a single phase. Correct shares are classified as those shares which are authenticated by at least $(t+1)$ different MAC's. As the authors considered $(0, \delta)$ security the protocol reliability fails with a small probability.

The protocol can be transformed to a $(0, 0, \gamma)$ protocol by simply carrying out error detection on the correct shares in the same way as described in Section 3.2. The same decision as described in the presented error detection scheme will also be taken.

With this transformation, this protocol also becomes a one phase $(0, 0, \gamma)$ protocol for a single message with a communication complexity of $O(n^2)$. In effect this protocol is equivalent to the $(0, 0, \gamma)$ protocol presented in the previous section[2].

5 Efficient Perfectly Secure Message Transmission

We now turn our attention to two-phase perfectly secure message transmission. The protocol we present achieves perfectly secure message transmission of a single message with $O(n^2)$ communication complexity and transmission rate in polynomial time. This greatly improves on previous protocols [7,10] which achieve this with $O(n^3)$ communication complexity and transmission rate.

We first present the main idea of our initial protocol and proceed to describe the main techniques that will be used in the protocol. In Section 5.3 we formally present our protocol and then present the security and complexity proof.

5.1 Main Idea

As our protocol is a two phase protocol, like most two phase protocols, in the first phase the receiver will send random elements of the finite field to the sender.

At the end of the first phase, the sender will observe the received data and identify possible errors that may have occurred in the transmission of the first phase. Different types of errors may have occurred and these will be outlined in Section 5.2. These errors will be sent via broadcast to the receiver in the transmission phase of the second phase. The sender will also send via broadcast correcting information so that the random elements sent in the first phase will constitute shares of the secret shared message of the transmission.

At the end of the second phase, using the identified errors, the receiver is able to identify all wires which were active during the transmission of the first phase. Using the correcting information, the receiver is able to securely obtain the secret message of the communication. The receiver is able to do this as it can ignore the shares that correspond to the identified faulty wires.

5.2 Main Protocol Techniques

Broadcast. Broadcast will be used in the second phase of our protocol. When the sender broadcasts information, the sender will send the same information over all n wires which connect the sender and receiver. As the adversary is able to corrupt at most t of these n wires, the receiver correctly receives the information via a majority vote.

[2] Using other authentication codes it is trivial to lower the transmission complexity.

Transmission of Random Elements. We now describe how random elements are sent from the receiver to the sender in the first phase of the protocol [3]. For simplicity we first describe the transmission of one random element r.

The receiver constructs the shares of r using a $(t+1)$-out-of-n secret sharing scheme. The receiver thus chooses a random polynomial p of degree at most t such that $p(x) = r + a_1x^1 + \cdots + a_tx^t$. The n shares $(s_1, s_2, \ldots, s_n)$ are computed by evaluating $p(x)$ at $x_1, x_2, \ldots, x_n$.

The receiver then proceeds to send share s_i via wire w_i $(1 \leq i \leq n)$. The receiver also transmits the t coefficients $(a_1, \ldots, a_t)$ and r - which define p, across a single wire.

In our protocol n parallel executions of the above will be carried out. n random elements $(r_1, \ldots, r_n)$ will be selected. The corresponding n random polynomials $(p_1, \ldots, p_n)$ will also be constructed. For each random element, using the corresponding polynomial n shares will be constructed. For reasons of clarity we denote as $(s_{i1}, \ldots, s_{in})$ the n shares for the i^{th} random element. Upon each of the n wires, n shares will be transmitted as will the definition of a single polynomial. The definition of polynomial p_i will be transmitted on wire w_i $(1 \leq i, j \leq n)$. The s_{ij} share constructed from this polynomial will be sent on wire w_j.

Error Detection and Identification of Faulty Wires. The above technique described what will occur and what will be transmitted in the first phase of the protocol. At the end of the first phase, the sender will receive n shares and a definition of a polynomial from each wire. Using this, the sender carries out error detection as follows.

For $i, j := 1, \ldots, n$ and for polynomial p_i received from wire w_i the sender considers the n shares received as the i^{th} share from each wire. The sender checks each of the shares and identifies as error shares the shares whose value does not agree with the definition of polynomial p_i. Share s_{ij} received from wire w_j is thus identified as an error share if $s_{ij} \neq p_i(x_j)$. The sender proceeds to broadcast the identified error shares to the receiver [4].

We now show that with this information the receiver can identify wires that were active in the first phase of the protocol. For clarity we assume that each error share is denoted as es_{ij} - with j indicating the wire w_j and i indicating the position from which the share was received by the sender $(1 \leq i, j \leq n)$. The i position of the share indicates that the share corresponds to the i^{th} polynomial received by the sender (from wire w_i).

The receiver checks the following cases to identify faulty wires:

Case 1: If the value of error share es_{ij} is different to the corresponding share sent out by the receiver in phase 1 then wire j is identified as a faulty wire.
Case 2: If the value of error share es_{ij} is equal to the corresponding share sent out by the receiver in phase 1 then wire i is identified as a faulty wire.

[3] This scheme is similar to the encoding scheme of Section 3.2. It also resembles techniques used in [14].

[4] This requires the sender to send a triple (j, i, v) to the receiver for each error share - indicating the i^{th} share of wire w_j with its value v. Alternatively if $n^2 \leq |\mathbb{F}|$ a pair of values (k, v) could be sent - where $k = j * n + i$.

Lemma 2. *The above cases correctly identify faulty wires of the first phase.*

Proof. Case 1: In Case 1, the error value received by the sender at the end of the first phase is identified to be different to the value sent by the receiver at the start of the phase. The only way this could have occurred is if the wire upon which the share was transmitted was actively controlled and the share was altered. The specific wire is thus correctly identified as a faulty wire.

Case 2: In Case 2, the value of the error received by the sender at the end of the first phase is identified to be the same as that sent by the receiver. The only way that a correct value of a share could be identified as an error by the sender is if it does not correspond to the corresponding polynomial. The only way this could occur is if the polynomial had been altered from its original form. The wire upon which the specific polynomial was sent is thus identified as a faulty wire. Following on from this we also prove the following lemma. □

Lemma 3. *If the adversary alters the polynomial transmitted across an adversary controlled wire, the specific wire will always be identified as faulty.*

Proof. As all polynomials are of degree at most t and as shares sent on honest wires cannot be altered, if the adversary alters a polynomial the maximum number of shares transmitted on honest wires that can be included on the altered polynomial is t. [5] This is a direct result from coding theory which states that polynomials of degree at most t can share at most t common points between them. As a result of this, there will always be at least one share transmitted on an honest wire which will be identified as an error by the sender at the end of the first phase. Following on from this, the adversary controlled wire will be identified as earlier described. □

5.3 Formal Protocol Description

We now formally present our protocol. For our protocol we assume the message of the transmission is $M^A \in \mathbb{F}$.

Step 1: The receiver does the following for $i, j := 1, \ldots, n$:
1. The receiver selects random element r_i.
2. The receiver constructs a $(t+1)$-out-of-n secret sharing scheme of r_i using the random polynomial p_i of degree at most t to obtain n shares $(s_{1i}, s_{2i}, \ldots, s_{ni})$.
3. The receiver sends polynomial p_i on wire w_i and share s_{ij} is sent on wire w_j.

Step 2: The sender does the following
1. The sender constructs a $(t+1)$-out-of-n secret sharing scheme of M^A to obtain n shares $(m_1, m_2, \ldots, m_n)$.

[5] The adversary can trivially carry out this alteration as the adversary knows the polynomial definition and thus all the shares.

2. For $i := 1, \dots, n$ the sender receives polynomial p_i from wire w_i. The sender evaluates $p_i(0)$ as r_i. The sender calculates the value $d_i := r_i \oplus m_i$. These are termed correcting information.
3. For $i := 1, \dots, n$ using the i^{th} shares received from each wire, error shares are identified. s_{ij} received from wire w_j is an error share if $s_{ij} \neq p_i(x_j)$.
4. The set of all identified error shares is sent to the receiver via broadcast.
5. The set of correcting information - $(d_1, d_2, \dots, d_n)$, is sent to the receiver via broadcast.

Step 3: The receiver does the following:

1. The receiver uses the technique of Section 5.2 to identify the set of active wires of the first phase. The set of honest wires (indicated as $HONEST$) is also constructed.
2. Using $HONEST$ the receiver computes shares of the secret message M^A. This is done by computing $m_{w_i} := r_{w_i} \oplus d_{w_i}$ where $w_i \in HONEST$.
3. Using the computed shares from the step above, the receiver interpolates and obtains the secret message.

5.4 Security and Efficiency

Theorem 2. *The above protocol achieves perfectly secure message transmission ($(0, 0, 0)$-security).*

Proof. We first prove the perfect privacy of the protocol. As the secret message is secret shared using a $(t+1)$-out-of-n secret sharing scheme and as the adversary is t-bounded, the adversary can only learn a maximum of t shares. This is because only t of the random elements received by the sender in Step 2 are learned by the adversary. These are the random elements whose polynomial definitions were transmitted on adversary controlled wires (these may have been altered by the adversary). The remaining $t + 1$ random elements are not learned by the adversary. This is because all random elements are secret shared using a $(t + 1)$-out-of-n secret sharing scheme and the adversary only learns t shares of each one. As a result, the adversary can only learn t shares of the secret shared message. Perfect privacy is therefore achieved.

Perfect authenticity of the protocol is achieved as the receiver only considers shares of the secret message whose corresponding random element was correctly received by the sender. This is achieved using the technique of identifying faulty wires described in Section 5.2. As shown, if the adversary alters the polynomial transmitted on an adversary controlled wire, the wire will always be identified as a faulty wire. Because of this, only correct shares are used in the reconstruction of the secret and thus perfect authenticity is achieved. Perfect availability is achieved as the receiver always accepts a message. The protocol is thus a perfectly secure message transmission protocol. □

We now analyze the complexity of the protocol. We denote as $|\mathbb{F}|$ the bit length of the field elements, $COM(1)$ and $COM(2)$ the communication complexity of the first and second phase of the protocol.

As the receiver in the first phase of the protocol sends n shares and a polynomial (defined by $t + 1$ field elements) across each wire the communication complexity of $COM(1)$ is $O(n^2)$.

The most expensive part of phase two in terms of communication complexity is the broadcast of the error shares identified in Step 2 of the protocol. As there are only $t+1$ honest wires, the minimum number of shares not identified as error shares by the sender will always be $(t+1)^2$. The maximum number of error shares is $n^2 - (t + 1)^2$. This is $O(n^2)$. Therefore, for the broadcast of the error shares $O(n^3)$ communication complexity is required. The communication complexity of $COM(2)$ is thus $O(n^3)$. As only one message is sent, the transmission rate of the protocol is $O(n^3)$.

It is easy to see that the computational costs of both sender and receiver are both polynomial.

5.5 2-Phase PSMT with $O(n^2)$ Transmission Rate

The protocol of Section 5.3 in its current form achieves $O(n^3)$ transmission rate. This is because of the $O(n^3)$ communication complexity of $COM(2)$ for the transmission of only one secret. We now describe how to decrease the transmission rate of the protocol to $O(n^2)$. The most expensive step in our two-phase protocol is the broadcast of the error shares by the sender to the receiver. The protocol is optimized by using a technique which allows for the reliable transmission of the error shares to take place with $O(n^2)$ communication complexity (as opposed to its current $O(n^3)$).

The technique of generalized broadcast was first presented in [14] and later used in [1,7]. The technique assumes the receiver knows the location of a number f of faulty wires. Generalized broadcast is then able to authentically transmit up to $(f + 1)$ field elements between a sender and a receiver using codes that can correct any (remaining) errors that may occur. This enables the authentic transmission of $(f + 1)$ field elements between a sender and a receiver with a communication complexity of $O(n)$ instead of $O(n^2)$ which allows for more efficient protocols. For a definition of generalized broadcast the reader is referred to Appendix A or [1,7,14].

As everything else in the protocol remains the same, the security proof of this version of the protocol remains the same as before. In the second phase of the protocol the sender uses generalized broadcast to transmit the error shares and broadcasts the n correcting information to allow for the secret message recovery on the receiver side. The second phase communication complexity - similar to the first phase, is now $O(n^2)$ and as only one message is transmitted so is the transmission rate of the protocol.

To the best of our knowledge this version of our protocol is the most efficient 2-phase polynomial perfectly secure transmission protocol for the transmission of a single message which exists in the literature. Previous efficient protocols included [10] and the protocol of Section 4 of [7].

6 Conclusion

In this paper we have introduced and formalized new security parameters to message transmission protocols.

We have also presented a polynomial protocol achieving almost secure message transmission for a single message. The protocol is of polynomial time and has $O(n^2)$ communication complexity. It remains an open question whether a polynomial time protocol with the lower bound of $O(n)$ communication complexity (as proven in [8]) can be created.

We have also presented a polynomial 2-phase perfectly secure message transmission protocol with $O(n^2)$ communication complexity for a single message. It would be nice to see if a more efficient protocol with lower communication complexity (for either one or both phases) could be achieved. Following on from this, it is also an open question to find a polynomial time protocol with linear transmission rate and a communication complexity lower than $O(n^3)$.

Acknowledgements. The authors would like to thank the anonymous referees for their valuable comments on improving the presentation of this paper.

References

1. Agarwal, S., Cramer, R., de Haan, R.: Asymptotically optimal two-round perfectly secure message transmission. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 394–408. Springer, Heidelberg (2006)
2. Dolev, D., Dwork, C., Waarts, O., Yung, M.: Perfectly secure message transmission. Journal of the ACM 40(1), 17–47 (1993)
3. Franklin, M., Galil, Z., Yung, M.: Eavesdropping games: A graph-theoretic approach to privacy in distributed systems. Journal of the ACM 47(2), 225–243 (2000)
4. Franklin, M.K., Wright, R.N.: Secure communication in minimal connectivity models. Journal of Cryptology 13(1), 9–30 (2000)
5. Jaggi, S., Langberg, M., Katti, S., Ho, T., Katabi, D., Medard, M.: Resilient network coding in the presence of Byzantine adversaries. In: INFOCOM, pp. 616–624. IEEE, Los Alamitos (2007)
6. Kumar, M.V.N.A., Goundan, P.R., Srinathan, K., Rangan, C.P.: On perfectly secure communication over arbitrary networks. In: PODC 2002, pp. 193–202 (2002)
7. Kurosawa, K., Suzuki, K.: Truly efficient 2-round perfectly secure message transmission scheme. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 324–340. Springer, Heidelberg (2008)
8. Kurosawa, K., Suzuki, K.: Almost secure (1-round, n-channel) message transmission scheme. In: Desmedt, Y. (ed.) ICITS 2007. LNCS, vol. 4883, pp. 99–112. Springer, Heidelberg (2009)
9. Patra, A., Shankar, B., Choudhary, A., Srinathan, K., Rangan, C.P.: Perfectly secure message transmission in directed networks tolerating threshold and non threshold adversary. In: Bao, F., Ling, S., Okamoto, T., Wang, H., Xing, C. (eds.) CANS 2007. LNCS, vol. 4856, pp. 80–101. Springer, Heidelberg (2007)
10. Sayeed, H.M., Abu-Amara, H.: Efficient perfectly secure message transmission in synchronous networks. Inf. Comput. 126(1), 53–61 (1996)
11. Shamir, A.: How to share a secret. Commun. ACM 22(11), 612–613 (1979)

12. Srinathan, K., Choudhary, A., Patra, A., Rangan, C.P.: Efficient single phase unconditionally secure message transmission with optimum communication complexity. In: PODC 2008, p. 457 (2008)
13. Srinathan, K., Kumar, M.V.N.A., Rangan, C.P.: Asynchronous secure communication tolerating mixed adversaries. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 224–242. Springer, Heidelberg (2002)
14. Srinathan, K., Narayanan, A., Rangan, C.P.: Optimal perfectly secure message transmission. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 545–561. Springer, Heidelberg (2004)
15. Wang, Y., Desmedt, Y.: Perfectly Secure Message Transmission Revisited. In: Knudsen, L. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 502–517. Springer, Heidelberg (2002)

Appendix

A Generalized Broadcast

In this section we describe generalized broadcast and its use in the protocol of Section 5.5. Generalized broadcast is a technique which combines broadcast and error correcting codes thus achieving a more efficient broadcast of the error shares. As mentioned in Section 5.5 generalized broadcast is used to decrease the communication complexity of phase two of the protocol from $O(n^3)$ to $O(n^2)$.

As generalized broadcast only concerns the second phase of the protocol, we assume that phase one has completed and the sender has identified all error shares that may have occurred. We now describe the further steps the sender carries out in order to transmit the error shares more efficiently.

The sender first defines the undirected graph G_e. An edge of the graph represents an error share that has been identified by the sender. The two vertices of an edge are the two wires involved with the error share - the wire from which the share was received and the wire whose polynomial definition the share is meant to correspond to. What is important to note here is that each edge of G_e always involves at least one faulty wire. This is because two honest wires can never cause an error share as no alterations occur on honest wires.

Definition 2. *A matching M of a graph $G = (V, E)$ is a set of pairwise non-adjacent edges. This means that no two edges in M share a common vertex. A maximum matching of a graph G is a matching that contains the largest possible number of edges.*

The sender proceeds to compute a maximum matching M_{G_e} of G_e. Denoting as M_s the size of M_{G_e} this indicates that there are M_s number of edges in M_{G_e}. It should be noted that $M_s \leq t$. Because of this, the sender is able to broadcast the maximum matching M_{G_e} of G_e to the receiver with $O(n^2)$ communication complexity. For each edge of M_{G_e} the sender will broadcast the received value es_{ij} of the error share, the wire w_r from which the share was received and the wire w_a with which it is associated with. As every edge in G_e (and thus in M_{G_e}) always involves at least one faulty wire, this allows the receiver to identify M_s

number of faulty wires (in the same way as described in Section 5.2). What is important for the encoding and transmission of all the error shares is that the sender is aware of this.

Suppose that the sender wants to send M_s number of elements $(e_1, \ldots, e_{M_s})$ to the receiver. The sender finds a polynomial p of degree at most M_s such that $p(1) = e_1, \ldots, p(M_s) = e_{M_s}$. The sender computes $p(M_s + i)$ and transmits the value on wire w_i where $1 \leq i \leq n$. The receiver in turn will receive n shares of an $(M_s + 1)$-out-of-n secret sharing scheme. This kind of code has a minimum Hamming distance of $n - M_s = 2t + 1 - M_s$. As $M_s \leq t$ this code does not allow the receiver to correct the maximum number of errors that may occur. However, as the receiver knows M_s number of faulty wires - through the broadcast of M_{G_e}, the receiver can ignore all shares received from these wires. The shortened code the receiver now considers is an $(M_s + 1)$-out-of-$(n - M_s)$ secret sharing scheme with a minimum Hamming distance of $n - M_s - M_s = 2t + 1 - M_s - M_s = 2(t - M_s) + 1$. This kind of code allows the receiver to correct $t - M_s$ number of errors which also equates to the number of faulty wires that have yet to be identified. The use of this shortened code thus allows the receiver to correct all remaining errors that may occur, reconstruct the same polynomial p and with perfect authenticity obtain the M_s elements $(e_1, \ldots, e_{M_s})$. With the transmission of n elements (one share per wire) the sender is thus able to communicate with perfect authenticity M_s elements to the receiver.

Following on from the above, as the size of the maximum matching is M_s this means that $2M_s$ vertices (which correspond to wires) appear in M_{G_e} and G_e. As at most n error shares can be associated with each wire the number of error shares that will need to be transmitted to the receiver are at most $2M_s n$. As shown, M_s elements can be transmitted using n elements and as there are at most $2M_s n$ error shares to transmit this can be carried out in $2n$ independent executions of the generalized broadcast method.

All of the error shares can thus be communicated from sender to received with perfect authenticity with $O(n^2)$ communication complexity. Because of this, the communication complexity of the second phase of the protocol is now reduced from $O(n^3)$ to $O(n^2)$.