Available online at www.sciencedirect.com
SCIENCE DIRECT®
Electronic Commerce Research and Applications 5 (2006) 170–188
Electronic Commerce Research and Applications
www.elsevier.com/locate/ecra

Design and implementation of real-time betting system with offline terminals

Panu Hämäläinen [a,*], Marko Hännikäinen [a], Timo D. Hämäläinen [a], Riku Soininen [b], Risto Rautee [b]

[a] Tampere University of Technology, Institute of Digital and Computer Systems, P. O. Box 553, FI-33101 Tampere, Finland
[b] Oy Veikkaus Ab, FI-01009 Veikkaus, Finland

Received 29 March 2005; received in revised form 12 September 2005; accepted 20 December 2005
Available online 20 March 2006

Abstract

Even though enabling electronic betting as an expanding entertainment field, the advanced communication and e-commerce technologies have not changed the nature of betting itself. This paper presents the novel offline Real-Time Betting (RTB) system overcoming the online processing and scalability limitations. It changes betting services into interactive sessions by supporting short, unpredictable target incidents as well as crediting winnings in real-time. The security design specifies reliable methods for time-stamping, storing bets, and verifying the authenticity of operations. A prototype system has been implemented for WLAN and DVB environments. Its performance and usability are evaluated and compared to traditional online systems.

Keywords: Real-time betting; Electronic betting; Offline; Security; Synchronization; Interactive systems; Entertainment; Wireless communications

1. Introduction

Advanced mobile terminals, multimedia networks, as well as e-commerce technologies have enabled numerous new applications for business and pleasure. Among others, betting organizers have begun offering their services through electronic systems, especially on the Internet. Instead of placing bets in front of a retail operator at a specific location, customers can attend betting remotely – virtually anywhere – with their own PCs, laptops, mobile phones, and set-top-box devices. The setup time and effort have markedly decreased and bets can be placed nearer the beginning of an event. In addition, winnings are credited online.

While the service has become more convenient for the customers, the fundamental nature of betting itself has not changed. The targets in electronic systems are still the same as in manual systems, and often provided in both ways. 'Will this penalty kick result in a goal' is an example of an unsuitable incident in a soccer match. There is not enough time or processing resources in user terminals, networks, and servers for handling such a bet. In order to change also the characteristics of betting from a static, slow procedure into an entertaining, interactive session that lasts throughout the target event, research on Real-Time Betting (RTB) has been carried out at Tampere University of Technology, Finland. The research has especially focused on providing betting with new characteristics over wireless user terminals, such as mobile phones, Personal Digital Assistants (PDA), and set-top-boxes.

Opposed to existing online systems, the developed system, referred to as *offline RTB*, does not require uplink communication from users to the organizer during the betting session. Instead, the user terminals only receive

* Corresponding author. Tel.: +358 3 3115 11; fax: +358 3 3115 4561.
E-mail addresses: panu.hamalainen@tut.fi (P. Hämäläinen), marko.hannikainen@tut.fi (M. Hännikäinen), timo.d.hamalainen@tut.fi (T.D. Hämäläinen), riku.soininen@veikkaus.fi (R. Soininen), risto.rautee@veikkaus.fi (R. Rautee).

1567-4223/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.elerap.2005.12.003

broadcast transmission from the organizer server and locally time-stamp and store the placed bets. New reliable methods are required for ensuring the authenticity of the operations during the session, especially at the user devices. The solutions presented in this paper allow implementing the user terminals as dedicated betting devices. Alternatively, the support for the offline RTB can be added to the existing wireless consumer devices, e.g., as a standard peripheral card.

Offline RTB achieves two fundamental goals for enabling interactive betting as a large-scale e-commerce application. First, it supports frequent bets on quick incidents and gives instant feedback when the incident outcomes are known. Second, the offline architecture is scalable, as it does not limit the number of betting terminals attending a session, while supporting the fast-phased betting. Offline RTB places only reasonable processing requirements for the system components.

The paper is organized as follows. In Section 2 the related work in the field is introduced. The concept of RTB and the requirements for a RTB system are defined in Section 3. Section 4 presents simulation results showing that current online methods with the existing user terminals cannot fulfill the requirements placed on RTB. Section 5 introduces the novel offline RTB architecture. Section 6 describes the offline RTB user terminal design. Suitable terminal and network technologies for an offline RTB implementation are discussed in Section 7. A prototype implementation for Wireless Local Area Network (WLAN) and Digital Video Broadcasting (DVB) environment with PDAs, PCs, laptops, and set-top box devices is presented in Section 8. The section also includes the results of real end-user experiments. Finally, Section 9 concludes the paper.

2. Related work

In traditional betting systems users place their bets in front of a retail operator. It is easy to take care of timing aspects: the betting counter is closed before the target event begins. The winnings are paid out after the event is finished. The practice for electronic betting is depicted in Fig. 1. There are numerous companies providing these so-called online services on the Internet. The betting terminal is a PC or a mobile phone. Also, dedicated devices operated by an authorized betting agent are used. The main server and duplicated bet databases are located in the organizer premises and a supervising third party is included for controlling the authenticity of operations.

In an online system users place bets with terminals, which transmit them to the main server. At the closing time the server stops accepting bets. When the outcome of the incident is clear, users may collect their winnings. Traditionally, the money reserved for a bet is paid to the organizer after choosing a bet target and winnings are paid out to a person presenting a winning voucher. Users may also have dedicated betting accounts for deducting stakes and crediting winnings.

In traditional online systems bet placements have to be completed early, before the target event begins. Bets are usually placed on the result of a complete match, race, or period. In addition, the availability of a certain bet incident with its exact opening and closing times is accurately defined in advance. A user placing a bet easily looses her excitement if incident outcomes become obvious already at the beginning of the event. The user must decide before the event starts whether or not to place bets, on which targets, and how much money to invest. If an incident is very short ('will team Finland win the opening face off'), it probably has a small winning ratio. Thus, users choose not to participate.

2.1. New commercial systems

At the end of year 2002, Fintoto in Finland released a tote service on the Internet. It enables betting on horses online, starting from few days before until the start of a heat. The betting server is prepared for the high peaks during the last few minutes of an open bet. It is capable of processing the maximum of hundred requests per second (10 ms per request). This setup is claimed to be sufficient for five years. Even though state-of-the-art technology is utilized, the system still provides traditional betting for a result of a heat [1].

Global Interactive Gaming (GIG) [2] has developed a real-time software betting system. The system enables betting on discrete parts of an event. However, it is also an online system, in which bets are collected before incidents

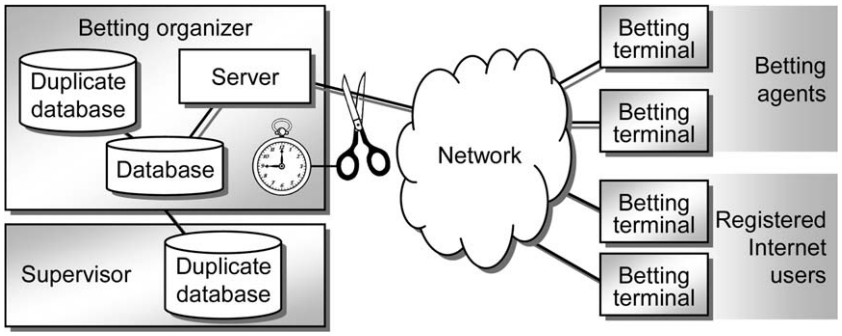

Fig. 1. General organization of electronic betting systems.

are settled. The main novelties are that the bet announcements are broadcast to terminals and the system constantly recalculates odds according to betting activity.

Oy Veikkaus Ab, a Finnish lottery company, has been developing online real-time betting service based on GSM Short Message Service (SMS). Several experiments have been carried out in 2003 and 2004. Short bets were introduced during ice-hockey and soccer matches. The tests have been extremely popular and the limit of a thousand volunteers has been reached instantly.

As the number of users increases and betting intervals shorten, the described systems require more processing resources (network throughput and server processing capacity) for maintaining the quality of service. Hence, they are not easily scalable.

2.2. Academic research

In addition to the commercial systems, academic research on the design of electronic betting and gambling has been carried out. The research has focused on making games and monetary transactions fair, secure, and verifiable for both users and organizers, not on real-time capabilities and scalability.

Protocols for electronic lotteries and casinos are described in [3]. The designs remove the requirement to trust the organizer without resorting to another trusted party. Instead, users are able to verify the correctness of execution. Similarly, [4] presents methods for users and organizers to prove and verify the validity of actions. In [5], a fair and verifiable gambling scheme is developed for devices with limited processing capabilities, such as mobile phones.

In [6] a gambling system consisting of an untrusted device and a smart card is defined. It can be used for gambling on randomized events (e.g. card games) without communicating with an organizer. The system does not support betting on time-limited events or on events the outcomes of which are defined by an external source, not by the smart card.

Zhao et al. [7] propose a gambling scheme for the Internet. The research concentrates on ensuring the fairness of games and payments and uses a trusted third party for resolving disputes. Sako [8] presents a server with which groups of users can create, participate, and verify their own lotteries. Similar lottery systems, in which the participants blindly influence on the values of the randomly-generated winning numbers, are developed in [9, 10, and 11].

While making the systems secure and verifiable, the solutions are not easily scalable since they still require two-way communications before the target incident is resolved.

3. Real-time betting

RTB is defined as an activity in which bet targets are continuously and frequently announced during an event and users try to predict the outcomes. After the outcome of an incident is known, users receive their winnings and invest them on new incidents. *Real-time* refers to that there is no limit for the minimum length of a bet, noticeable to users. A bet may be open only a second and the system is still able to handle it.

The components of a RTB system are depicted in Fig. 2. An *event* is a target happening which contains discrete *incidents* from which bet targets are chosen. An *operator* is the entity which follows the event, announces bets, and defines results. The announcements are transferred to a betting server via a secure channel. The server processes and stores bet and user information. It informs users about the announcements through another secure channel. A user follows the event and places bets with a betting terminal.

In RTB, a suitable incident must be a distinctly separable occurrence during an event. It must have an unarguable opening time, closing time, and discrete outcome. When the event itself is well predictable and sequential, bets are easier to define. For example, long jumping is an event having these characteristics. On the other hand, incidents in a soccer match are more complicated to distinguish. Commonly repeating bet types can be prepared as *templates* in advance, which makes it easier for an operator to react when potential incidents occur.

A key requirement in RTB is fairness. Each user should get her bets placed independently of the geographical location within the service coverage area. In addition, it should be possible to place a bet just before the closing time when most information on the probable outcome is available. How near the closing time the decision is made, should not significantly affect the bet acceptance probability. Fulfilling the fairness requirement is important for creating the excitement of RTB.

When the number of users is relatively small, existing online systems can be used for implementing limited RTB. The maximum number of users is set by server processing capacity and network throughput. Also, these define how short bets are possible. Fairness requires preparing for high peak traffic. Particularly, the service cannot meet the fairness requirement on the Internet since users with better connections are favored.

An advanced implementation consists of distributed, trusted time-stamping devices [12]. They communicate with the terminals appending placements with unchangeable timing information. At a suitable time, the betting server

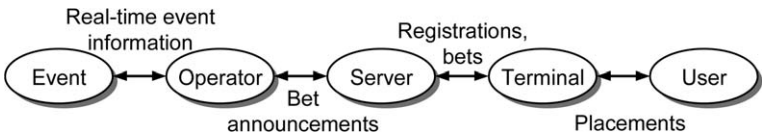

Fig. 2. Components of RTB system.

retrieves the data and computes the results. The main drawback for RTB is that, for balancing the processing load, the distribution of the devices has to be well-designed and the location of terminals known. For example, the devices by Symmetricom [13] can produce 50–125 protected time stamps per second, implying a large number of devices for large-scale RTB.

4. Online performance for RTB

In order to evaluate online system performance for RTB, a simulation model was developed in Specification and Description Language (SDL) [14]. SDL is a high-level description language for implementing entities on different abstraction levels with parallel extended finite state machines. The entities can be connected to form a single system model for design verifications and simulations. The design tools enable generating application (C code) from the SDL descriptions.

For allowing the free movement of users, wireless networks are preferred in providing RTB services. Being one of the wireless technologies with the highest capacity and the lowest delay, IEEE 802.11b WLAN [15] was chosen for the connection in the model. Several stadiums in Finland already contain WLAN access points with Ethernet backbone network. The medium access control protocol was implemented in the Distributed Coordinate Function (DCF) [15] mode.

A bet was placed by sending a 100-byte packet from a terminal to the server. If the transmission was successful, the server responded with a 100-byte acknowledgement after processing the request. A bet was regarded valid only if the server managed to acknowledge it before the closing time. In order to simulate real wireless environment with errors bursts, a discrete two-state Markov model was added to the channel [16]. The probability of transitioning from the good state to bad was 5% and 30% vice versa, taken from [17]. A transmission was successful in the good state and failed in the bad state.

Fig. 3 shows the results for two different simulations. 500 terminals started transmitting placements simultaneously at the beginning of Simulation 1. The WLAN access protocol ensured that the channel did not become jammed. The server processing time was 11.7 ms per bet on average, which is in line with the Fintoto system [1]. It was estimated to include all the network and processing delays outside the WLAN. Despite the heavy access contention and rather pessimistic error model, all the terminals were able to transmit within 0.57 s. However, the server became the bottleneck. Its processing queue started decreasing only after the terminals had finished transmitting. It took 5.9 s before the server had finished processing. Because of the growing queue, the later a terminal managed to transmit its bet, the longer it had to wait for the response. The last terminal had to wait for 5.3 s.

The bet placements of the 500 terminals were randomized in Simulation 2. The simulation was divided into small time intervals, during which a new terminal started attempting to transmit with 10% probability. The server processing time was 12.6 ms per bet. The other parameters were unchanged. The terminal transmissions were finished in 3.6 s. The maximum size of the server queue was lower

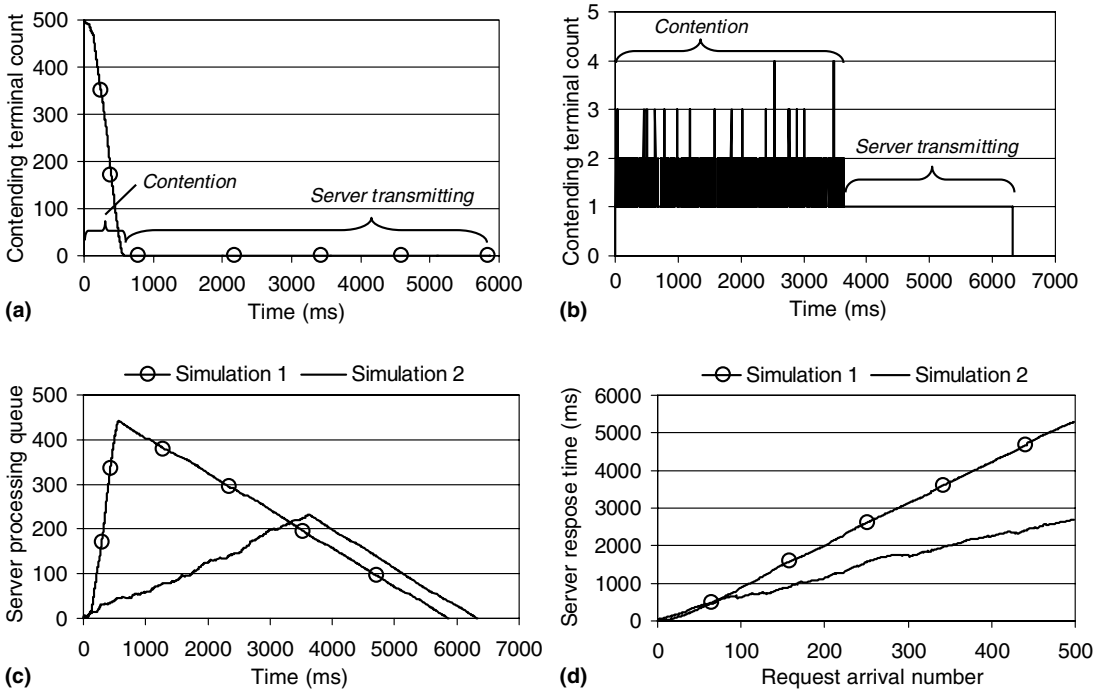

Fig. 3. Online betting simulations in WLAN: (a) the number of contending terminals as a function of time in Simulation 1 and (b) in Simulation 2, (c) server processing queue lengths as functions of time, (d) server response times as functions of the arrival number of a terminal request.

but, due to the randomization, the total processing time was about 0.5 s longer. Similarly to Simulation 1, the response time gradually increased as more bets arrived. At some points the server was able to process the requests faster than new ones arrived. Hence, the queue and response time did not grow as linearly as in Simulation 1. The maximum response delay was 2.7 s.

The WLAN simulations show that the most significant part in the response delay comes from the processing time. The terminals were able to transmit much faster than the server processed, even when all the 500 terminals started competing of the channel at the same time. The simulations illustrate that an online system with the used setup is not capable of providing RTB even for a reasonably small number of players. For example, if a bet was open for 5 s, the placements from the last 67 terminals in Simulation 1 and the last 112 terminals in Simulation 2 would be rejected. The tests with real audience in Section 8 will show that this is in fact the case: a decision to place a bet is often made during the final 10 s even if the bet is open for 30 s.

In Simulation 2, the terminals which placed their bets last did not get them through. As an opposite, in Simulation 1 the success was randomized by the WLAN contention. Thus, the later a bet is placed at a terminal or the more WLAN terminals are competing for the channel access, the higher the rejection probability is. These remarks on an online system break the fairness and real-time requirements earlier set for RTB.

The simulations did not take into account possible retransmission due to acknowledgement timer expirations. Because most terminals wait several seconds to receive a response, it is likely that their timers expire several times resulting in unnecessary traffic. This would significantly increase the network load as well as keep the server processing queue growing. By decreasing the processing time it would be possible to decrease the response time. However, increasing terminal count would eventually cancel the gain of increased processing power. Consequently, the online solution is not easily and inexpensively scalable.

For comparison with other common wireless networks, GSM and GPRS data transfer delays were measured. The connection was created through a modem pool to a destination computer in the same LAN as the pool. In the test a 100-byte packet was transmitted 1000 times. The average delay was 999 ms for GSM and 742 ms for GPRS. The delay in a WLAN via an access point was only 3 ms on average.

5. Offline architecture for RTB

In order to overcome the real-time processing problems of online betting systems, the novel *offline RTB* system has been developed. The key idea is to time-stamp and store bet records locally at terminals instead of transmitting them to a server. The server collects the records after the target event is finished and incident outcomes already revealed. The offline RTB avoids high processing peaks and supports scalable, fair electronic betting in short time cycles regardless of the number of participants and betting frequency.

The architecture is illustrated in Fig. 4. The data and broadcast networks can be separate or the same. The two-way data network is utilized for user registrations and bet record collections whereas the broadcast transmissions convey bet announcements. The term *offline* refers to the bet placement phase in which terminals only receive broadcast downlink data and locally store bet records generated from user inputs.

Participating betting in the offline RTB comprises of three stages: *registration*, *betting session*, and *bet record collection*. This section presents the general principles of the stages. Section 6 describes the security and timing related processing at a terminal in more detail.

5.1. Security challenges

The offline solution adds new constraints especially for terminal security and timing. Reliable methods are required for ensuring that accepted placements are made before outcomes are known and that they have not been changed afterwards. Due to the application, the operation environment of a terminal is considered hostile. It is required that the most critical parts of security and timing

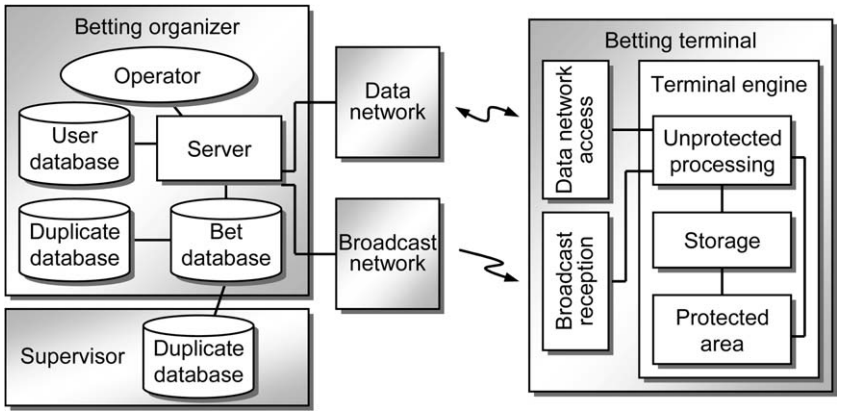

Fig. 4. Offline RTB architecture.

operations are protected from external parties, including users. Thus, the RTB terminal includes a *protected area* as depicted in Fig. 4. Only an authentic server can access it through the data and broadcast links. Unauthorized access must require much more effort and expenses than it can pay off.

Local time-stamping requires that the terminal time is synchronized to the server time and that the synchronization is maintained throughout a betting session. The synchronization has to be protected from the intervention of external parties. Also, since bets are opened and closed via the broadcast link, a terminal has to receive the broadcast in time. A user should not be able to postpone the closing time of a bet or modify the terminal time. These would allow betting on an incident after its outcome is already known. The server and/or the terminal must be able to recognize if the synchronization or the timing of the broadcast is tampered with. On the other hand, reasonably fixed, standard network delays should not prevent operation.

In order to guarantee that only an authorized user has access to the betting account and can attend betting, the user has to be reliable identified at the server during the registration. In addition, the two-way data link must be protected from outsiders. Similarly, it must be possible to verify the origin of the broadcast to disable packets from a fraudulent third party. A user should not be able to falsely deny placing bets.

5.2. Registration

Registration is possible before an event begins and also any time during the event. The two-way communication of the registration is depicted in Fig. 5.

In order to create a connection, the user is authenticated with the server (*user authentication*). After the authentication the user gets access to her account and the lists of upcoming events. Event information includes an event identifier, description, start time, types of target incidents, and a limit for acceptable stakes. Once an event has been chosen, the terminal transmits the server a *registration request*. The request consists of the event identifier and the amount of money to be reserved. The monetary transactions are carried out as in online systems. The reserved sum is deducted from the account and stored in the terminal with the rest of the registration information.

To guarantee valid operation during the betting session, the protected area and the server have to agree on certain parameters secretly from the user. Upon the reception of the registration request, the server initiates an authentication procedure with the protected area (*terminal authentication*). Next, the server transmits parameters for verifying the validity of the broadcast transmission. The server also requests the protected area to synchronize its wall clock time to the server time and to initialize its local storage for storing data during the betting session. Finally, the terminal closes the two-way link and starts waiting for the event to begin.

5.3. Betting session

The betting session utilizes only the broadcast link. Each broadcast packet contains data for both the protected and unprotected areas. The unprotected area contains the betting application and the protected area ensures the validity of the betting session. The RTB session communication is illustrated in Fig. 6. Upon the reception of a broadcast packet, the unprotected area copies and forwards it unchanged to the protected area. The protected area verifies the authenticity and timing of the packet. The unprotected area is informed about the result with a *status* message. If the packet is valid, the unprotected area continues processing it.

The broadcast packet types and contents area shown in Fig. 7. All of them contain protocol, event, and packet identifiers (ID), size, type, and a server time stamp and signature. The event, packet, bet, and tick IDs are sequence numbers. The server ensures that each event has a unique event ID and each packet, bet, and tick ID is unique during an event. The open, close, and result packets of a bet share

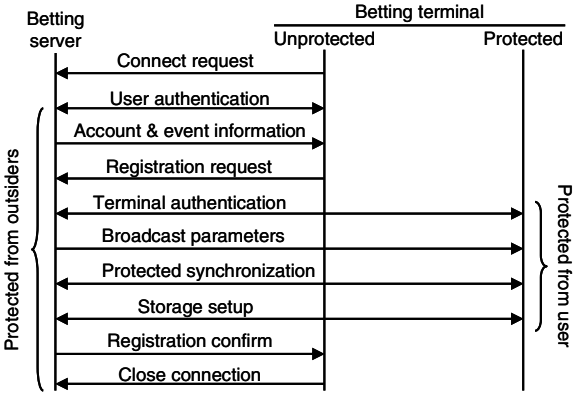

Fig. 5. Offline RTB registration.

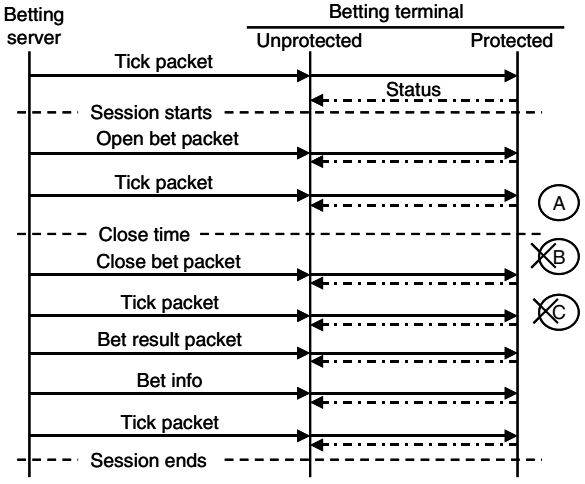

Fig. 6. Offline RTB session.

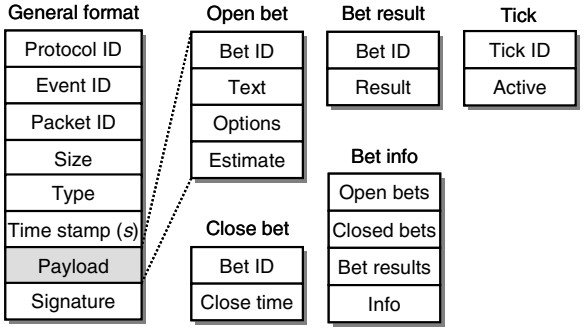

Fig. 7. Broadcast packet formats.

the same bet ID. Before a target event begins, the server starts transmitting periodic *tick packets*, which are used for timing. The *active* field defines whether the event has started or not.

As a suitable incident emerges, the operator opens a bet at the server and an *open bet packet* is broadcast. It contains a bet ID, textual target information, result options, odds, and an *estimate* of available betting time. The user may choose a stake and place a bet. The placement information is given to the protected area for time-stamping.

Before the outcome is settled, the operator sends a *close bet packet.* It contains *close time*, a time instant after which placing a bet for the incident was not allowed. Already the close bet packet disables placing the bet at a terminal (placement C in Fig. 6). In addition, the protected area verifies that the bet was placed before the close time (placement A). This *guard period* ensures that the broadcast delay, the slow reaction of the betting organizer operator, and synchronization inaccuracies do not allow placing bets after the outcome of the incident is already known (placement B). The protected area informs the unprotected area with a status message whether a placement was accepted.

When the outcome is clear, the terminal receives a *bet result packet*. The unprotected area compares the received result with the user's prediction. If they match, it adds winnings to the local balance. *Bet info packet* is a periodic information packet. The *open bets list* of the packet contains the bet IDs of last few open bets. Similarly, the *closed bets list* includes the bet IDs of last few closed bets and their close times. The *bet results list* carries the results of last few bets. The lists allow users to stay up-to-date even if they miss bet packets. An operator may transmit additional information to users in the *info* field.

The betting continues until the active field in tick packets is cleared or the user decides to stop betting. A terminal continues receiving tick packets until betting is stopped.

5.4. Bet record collection

The bet record collection utilizes the two-way data link. At a suitable time, the server reuses the previously negotiated data link session or reinitiates the user authentication. It requests the terminal to transmit its stored betting session data consisting of the bet records and the trace of the session. The session data are stored in the unprotected storage shown in Fig. 4. As described in Section 6, they are cryptographically protected with the keys of the protected area. Hence, the session data are not compromised in the storage or during the transmission to the server.

The server checks that the data are valid and computes the final outcome for the session. Winnings are added to the betting account and the user is informed about the final result.

6. Offline RTB terminal design

The betting terminal plays the key role in ensuring the authenticity of the offline RTB session. In this section the design of the offline RTB terminal is presented. The security rationale for the design choices is given in each subsection. Many of the utilized components, such as the cryptographic algorithms, are well-known as well as widely used and trusted in other contexts. According to the authors' knowledge, in this work they are combined and utilized in a novel way and for a new application purpose. In addition, the methods for ensuring the secure timing at the terminals using the broadcast and combining them with the time-stamping of the protected storage are novel.

Compared to existing distributed systems, the time synchronization problem is different in the offline RTB. Usually, the user wishes the synchronization to be as accurate as possible. On the contrary, in the offline RTB a malicious user wants the synchronization to be inaccurate so that bets could be placed after the results of target incidents are available. The offline RTB also changes the synchronization requirements and possibilities from existing systems. The absolute long-term synchronization accuracy does not have to be very high since the synchronization has to be maintained for a fairly short period of time. In addition, the timing resolution can be moderate as only the timing of actions performed by human users is considered. However, the synchronization should still be rigid and respect certain limits for the trustworthiness of betting sessions.

The broadcast synchronization methods of common technologies, such as Global Positioning System (GPS), the National Institute of Standards and Technology (NIST) time service [18], and the de facto computer network synchronization standard Network Time Protocol (NTP) [19], are not suitable for the offline RTB betting sessions, although their accuracy can be very high. Typically, in those systems the synchronization is a continuous process that lasts throughout the operation time of a device. Contrarily, in the offline RTB the synchronization can only be performed before the target event begins, as explained in Section 6.2.4.

The functional entities of the offline RTB terminal are shown in Fig. 8. The protected area contains components for broadcast security, terminal data security, storage management, protected time control, tick control, and a counter. The unprotected components are network and user interfaces, user data security, game engine, and data storage.

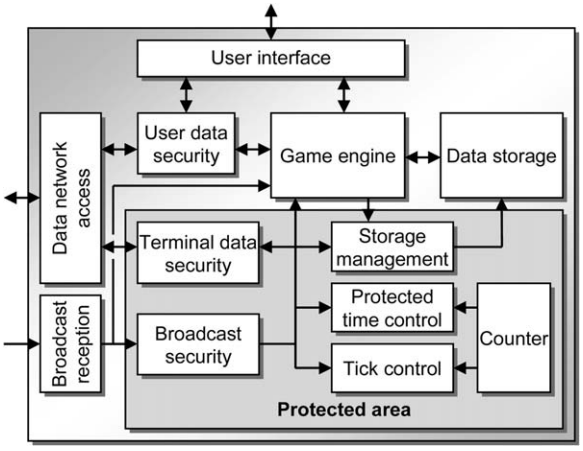

Fig. 8. Offline RTB terminal architecture.

The game engine in the unprotected area contains the functionality of the betting application. It communicates with the user data security entity, User Interface (UI), data storage, broadcast interface, and protected area. The game engine generates bet records from user inputs, computes outcomes, updates the balance, and returns the session data to the server. The protected area makes decisions about bet placement acceptance, verifies the validity of terminal operations, controls the authenticity and timing of broadcast, time-stamps bet records, and creates a complete, protected session trace.

In the following, the applied cryptographic algorithms and communication protocols are considered secure, meaning that either they have been proven secure or it is not known how to break them. For example, it is assumed that it is not possible to find a collision for the utilized one-way hash, fabricate a digital signature, or eavesdrop on message exchanges of a protected link. The assumption is typical in security designs in which a specific cryptographic algorithm is not important but rather how and for what purpose it is used. If the properties of an algorithm are not found sufficient, it can be switched to another one. A similar assumption is often made in theoretical cryptography, e.g., about the underlying algorithm when proving a security bound for a data authentication scheme [20].

6.1. Communications

The protected and unprotected areas share the two-way data link by creating their own logical connections to the server. It is required that both the connections are mutually authenticated and fresh session keys are generated between the communicating parties. In addition, the utilized security protocol must ensure the confidentiality, freshness, and authenticity of the communications following the authentication. Commonly trusted secure channels, such as the Transmission Layer Security (TLS) protocol [21], are suitable for the purpose.

The *user data security* entity performs the user authentication over the data link. The mutual authentication is required for preventing the users from revealing their login information to a fraudulent party and for ensuring that the server only allows authentic users to access their accounts and register to betting sessions. The authentication generates a shared master key K_m between the unprotected area (user) and the server. The session key K_s is computed using a cryptographic hash algorithm (H) for the key separation

$$K_s = H(\text{“SessionKey”}|K_m), \quad (1)$$

where | stands for concatenation. For instance, SHA-256 [22] is a suitable hash algorithm. After the authentication the connection is protected with K_s. Another key is also computed from K_m in Section 6.3.

The user authentication allows only communicating with the unprotected area. The authentication can be based on user name/password pairs, public key certificates, as well as smart cards. The TLS handshake [21] using both the server and client side certificates is a valid method. Another suitable technique is the Secure Remote Password (SRP) protocol applied for the TLS authentication [23]. SRP provides password-based mutual authentication and session key generation without requiring Public Key Infrastructure (PKI).

The communication between the protected area and the server is protected in the same way by the *terminal data security* entity. In order to decrease the processing requirements of the protected area, an authentication protocol that does not require public key operations is preferred. Kerberos [24], which has been widely employed and analyzed, is such an authentication system. The authentication material is pre-installed in the protected area. The fresh session key generated during the authentication must be long (128 bits) and it is used for protecting the protected area initialization from outsiders, including the user.

In contrast to the two-way link, broadcast packets presented in Fig. 7 carry data for both the protected and unprotected areas. As the packets do not contain any confidential information, it is enough to ensure only the authenticity of the data, which is provided with a digital signature in each packet [25].

The broadcast packets are processed by the *broadcast security* entity of the protected area. In the registration the server transmits its public key to the entity for verifying the authenticity of the broadcast. The signing key is only valid for a single betting session. Therefore, the signature does not have to be extremely strong, which results in lower processing requirements. During a betting session an invalid signature does not trigger any other actions except for dropping a packet. However, it may later disable the operation due to failing in the timing checks of the following sections.

Alternative, the signatures can be verified in the unprotected area, which further decreases the cost of the protected area. This is enough for convincing the user that the broadcast is coming from an authentic source. However, as the protected area and the server cannot trust the unprotected area, the protected area must store the

complete contents of each received broadcast packet into its betting session trace. When the trace is returned to the server, it can verify that the received broadcast data are the same as the transmitted data. The drawbacks of the solution are the increased storage size and the fact that broadcast tampering by the user is only detected after the end of the session.

The broadcast security entity also verifies that no packets have been missed or replayed. This is carried out by checking the event ID, packet ID, and tick ID fields. The event ID is unique and a packet or tick ID should always be one greater than the previous one. Packets with passed IDs are dropped. Packets with higher IDs are accepted but the actions of Section 6.5 are performed.

6.2. Timing

The procedures of this section address maintaining a reliable relationship between the time of the protected area and the time of the server and detecting attacks on timing. An attack against these procedures is considered successful if an attacker (a malicious user) manages to place a valid bet after the result of a target incident is known. For a successful attack, the human attacker must have enough time to first observe the result of the incident and then place the bet.

6.2.1. Protected counter

All the time-related processing in the protected area is based on the protected *counter*. The main purpose of the counter is to provide time stamps for bet records. The requirements are that the counter is monotonic, relatively stable and accurate (its drift rate from the server clock is small during a betting session), it does not roll over between the registration and the bet record collection, its precision is at sub-millisecond level, and its frequency cannot be significantly affected by changing the operating conditions (voltage, temperature, electromagnetic fields etc.). In order to transform counter cycles into seconds, a fixed mapping value (frequency) f is stored in the protected area. The server requests and stores it in the registration for the final verifications of the bet record collection.

The counter is implemented in hardware. Preferably, the protected area contains several parallel counters implemented in different technologies, which makes it extremely difficult to modify all of them in the same way. The rates of the counters must remain constant relative to each other. The drift rate requirement is not very strict as the counter is used for timing actions performed by human users and the length of a betting session is typically limited to a few hours. An attacker can only attempt exploiting the counter inaccuracy if its drift is negative from the server clock. A positive drift only means that the user has less time for placing a bet.

In the following sections the counter drift rate is referred to as u and the time the counter has been drifting as T (i.e. T is the time elapsed from the previous synchronization). For example, with a typical oscillator stability of ± 15 ppm (parts per million) the rate u is ± 15 ns per second. When T is 5 h, the total drift is at most ± 270 ms, which is definitely too little for a successful attack. Oscillators with much better stabilities (0.5 ppm) are generally available [26]. Because of the guard period, bets are also closed earlier, e.g. 1 s before, than exactly at the time of the result resolution and much larger errors can be tolerated.

The *tick control* entity employs the counter and the tick packets for controlling that broadcast transmission is actually received and it is not delayed. Each tick packet has a predefined time window, inside which it must be received. Even though it is difficult to tamper with the protected area, the tick packet verifications are used for detecting changes in the counter rate.

The counter is used by the *protected time control* entity for maintaining an estimate of the server wall clock time during the betting session. The estimate is required for time-stamping bet records and verifying the timing of broadcast packets, which prevents recording and replaying a complete session. The entity also controls that the error of the wall clock estimate does not exceed set limits. The timing parameters of a betting session are defined in Table 1.

6.2.2. Counter verifications

The relations of tick packet parameters are presented in Fig. 9. The server defines an interval b for successive ticks during the registration. When the protected area receives its first tick after the registration stage, it records the counter value of the reception time r_0. Then, the tick control computes an estimate for the reception time of the next tick packet

$$r_i = r_{i-1} + fb, \tag{2}$$

Table 1
Offline RTB timing parameters

Parameter	Unit	Description
b	seconds	Pre-defined reception interval for tick packets
c_i	cycles	Read counter value
d	seconds	Minimum broadcast delay
e_i	cycles	Measured error between r_i and c_i
f	Hz	Fixed protected counter mapping
g	seconds	Allowed deviation for the broadcast time stamps and the terminal wall clock estimate
h	seconds	Allowed error for the reception times of tick packets
m	seconds	Minimum data link round-trip delay
o	seconds	Offset of the wall clock estimate
p	seconds	Limit for the data link round-trip delay
r_i	cycles	Precomputed reception time for tick i
s	seconds	Wall clock time stamp of the server
t	seconds	Terminal estimate of the server wall clock
T	seconds	Time elapsed from the previous synchronization
$T_{\max}$	seconds	Maximum allowed value for T before betting session
u	ppm	Protected counter drift rate
v	seconds	Standard broadcast delay variation

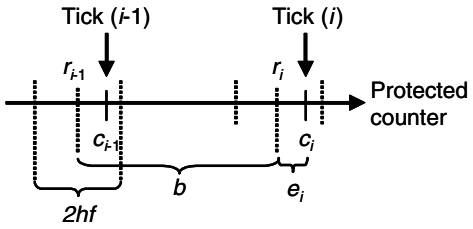

Fig. 9. Tick packet timing.

where $i = 1, 2, \ldots$ is the sequence number of the received tick. Thereafter, upon the reception of the next tick packet, an error value is computed

$$e_i = |r_i - c_i|, \tag{3}$$

where c_i is the real, read counter value at the reception time. The tick control compares

$$e_i < hf, \tag{4}$$

where h is the allowed error in seconds between the real and the estimated tick reception times defined in the registration by the server. If the condition does not hold during a betting session, it is deduced that either the counter rate has changed or part of the broadcast transmission is intentionally delayed. A violation is recorded (Section 6.5). If the condition of Eq. (4) fails a certain number of times before the event has started, resynchronization is triggered as described in Section 6.2.4, the value r_0 is refreshed, and computing Eq. (2) is restarted.

The size of the error limit h depends on the quality of the utilized broadcast channel. If the channel has a well-defined, constant standard delay, h can be small and already small violations are detected. However, because the reception time of the first tick packet r_0 is used as the reference point for the rest of the ticks, the limit h should be larger than the standard delay variation v of the broadcast channel. Otherwise, if the standard delay of the first tick packet is coincidently at the maximum level, the next packets may not satisfy Eq. (4), even if their delay was also inside the standard variation. By choosing $h = 2v$, this situation cannot occur, as clarified in Fig. 10.

After the first tick, the broadcast transmission can intentionally be delayed at most by $h + uT$. This means that in addition to h, an attacker can cumulatively delay the broadcast at most by uT. The term *cumulative delay* refers to the total delay that is constructed by delaying each packet with a small, non-violating amount. With the example of Section 6.2.1 the maximum cumulative delay after five hours is only 270 ms, which is not enough for a successful attack. The cumulative delay is limited to the counter drift because the reception time estimates (r_i) are computed from the reception time of the first tick after the synchronization. An exploitable delay could be caused if the next estimate was always calculated from the real reception time of the previous tick (c_{i-1}). This would enable postponing the broadcast by $nh + uT$, where n is the number of received tick packets. For example, if tick packets were transmitted in 30 s intervals and $h = 20$ ms, after 5 h the cumulative delay would be over 12 s, regardless of the counter stability.

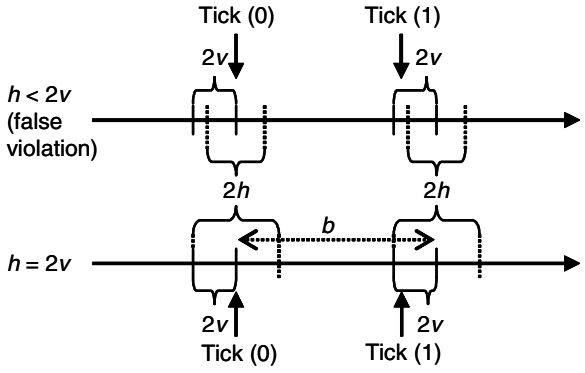

Fig. 10. Choosing the limit h for tick packets.

The verification of Eq. (4) does not trigger countermeasures if the complete broadcast is delayed with a fixed offset (shifted), because it only verifies the tick reception times relative to each other, starting from the tick received first. This type of a complete session replay is prevented with the wall clock estimate methods described in the next two sections.

6.2.3. Wall clock estimate

The local estimate of the server wall clock time maintained by the protected time control entity is

$$t_i = t_{i-1} + f(c_i - c_{i-1}), \tag{5}$$

where $i = 1, 2, \ldots$ is the sequence number of the estimate update operation, c_i is the value of the protected counter at the time of performing the update, and $t_0 = c_0 = 0$. The update procedure is performed every time the estimate is read.

The wall clock estimate is synchronized to the server wall clock time in the registration through the two-way data link. In the basic method [27] the wall clock offset at a terminal is

$$o = \frac{(s_r - t_t) + (s_t - t_r)}{2}, \tag{6}$$

where t_t is the terminal wall clock estimate when transmitting a synchronization request, s_r is the server wall clock when receiving the request, s_t is the server wall clock when replying, and t_r is the terminal wall clock estimate when receiving the reply. For better results, the message exchange is performed several times. The offset is added to the wall clock estimate

$$t_n = t_n + o, \tag{7}$$

which is the value of the first wall clock estimate after the synchronization. The value t_n on the right side of the equation is computed according to Eq. (5). The offset o is also recorded by the server for the final session verifications.

For better accuracy, the two-way wall clock synchronization can utilize more advanced methods, such as NTP and IEEE 1588 [28]. For example, NTP estimates the dispersion, frequency error, and stability of the local clock as well as handles discontinuations. The newest NTP version also

includes cryptographic methods for authenticating the synchronization source [29]. In the offline RTB registrations these cryptographic features are unnecessary as the same protection is already provided by the created two-way connection.

Regardless of the used synchronization method, it is required that the protected time control reasonably accurately knows the network round-trip delay of the message exchange and discards messages exceeding a certain tolerance level. Otherwise an attacker can try to delay the messages to make the wall clock estimate loose time. Before the offset computations, the protected time control compares

$$(t_r - t_t) - (s_t - s_r) < p, \tag{8}$$

where p is the limit for the data link round-trip delay defined by the server before beginning the synchronization procedure. If the condition is not satisfied, the values are discarded.

The smaller p is chosen the better synchronization is achieved. However, p cannot be smaller than the real minimum round-trip delay, since in that case none of the message exchanges satisfies Eq. (8). A similar delay tolerance has been introduced in [30] in order to ensure precise synchronization in distributed systems. NTP also sets limits for the round-trip delay in its clock selection and adjustment algorithms. In this work the limit serves for two purposes: ensuring precise synchronization for benign users and limiting the advantage (offset error) a malicious user can achieve by delaying the synchronization message exchange. Assuming that the protected counter drift rate compared to the server clock is negligible during the message exchange, the maximum offset error over a symmetric link is

$$\frac{p-m}{2}, \tag{9}$$

where m is the real minimum round-trip delay [30].

If the counter satisfies its requirements, resynchronization is not required after the registration since a betting session is fairly short, typically few hours (cf. Section 6.2.4). For example, when considering $p = 10$ ms and $m = 3$ ms, which was the average round-trip delay for WLAN in Section 4, with the typical counter stability of Section 6.2.1 the wall clock estimate error is at most ± 273.5 ms after 5 h. This is still too little for enabling a successful attack.

6.2.4. Wall clock verifications

The protected time control entity uses the wall clock estimate and the server time stamps of the broadcast packets for verifying that individual packets arrive at the right time, bets are placed before closing times, and the complete broadcast transmission is not intentionally delayed, including the first tick packet. The relations of the wall clock verification parameters at the protected area are illustrated in Fig. 11.

In the registration the server defines the estimate for the minimum broadcast delay d and the allowed deviation g for the server time stamps of broadcast packets and the local wall clock estimate. A similar broadcast delay estimate

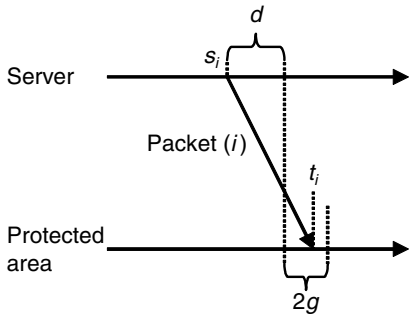

Fig. 11. Wall clock estimate timing.

can be used in NTP. The deviation limit g includes the standard broadcast delay variation v. Upon the reception of a packet, the protected time control compares

$$|t_i - (s_i + d)| < g, \tag{10}$$

where $i = 0, 1, 2, \ldots$ is the sequence number of the received packet, t_i is the value of the local wall clock estimate at the time of reception, and s_i is the server time stamp of the packet. If the condition is not satisfied during a betting session, the protected area performs the actions defined in Section 6.5. The verification detects packets that have been delayed more than $g + uT$.

In addition to limiting individual packet delays, the verification ensures that the broadcast transmission can intentionally be shifted at most by g. Here the coefficient uT is negligible because T is very small at the time of receiving the first tick. After the first tick the cumulative delay is limited as described in Section 6.2.2. Hence, the total intentional delay an attacker can cause without detection is limited to $g + h + uT$.

If the target event has not begun and Eq. (4) or Eq. (10) fails a server-defined number of times or T exceeds a server-defined limit T_{max} (e.g. 2 h), resynchronization is performed over the two-way data link. This ensures that betting is not denied from benign users during the upcoming session because of the instability of the protected counter. It also limits the advantage the counter instability can provide to a malicious user. The cumulative delay is limited to $u(T_{max} + w)$, where w is the elapsed time from the betting session start and the last synchronization was performed T_{max} before the start. Therefore, the maximum undetectable delay during a betting session is limited to $g + h + u(T_{max} + w)$. The resynchronization reuses the previously negotiated data link session or reinitiates the authentication procedures for a fresh connection and it is carried out as the original synchronization. The server records the new offset.

When a close bet packet is received, the protected time control entity verifies that the corresponding bet has been placed before the close time. The close time and the time stamp of the bet record are directly compared.

The wall clock verifications cover other timing attacks except the modification of the offset or the counter rate at the same time with the intentional delay of the broadcast. The offset modification with delayed broadcast is

detected by the counter verifications. The counter verifications can also be more precise since the limit h must only include the variation v. The limit g must contain both v and the allowed wall clock estimate error. In addition, the comparison of Eq. (10) includes the estimate d. The offset change is also detected in the bet record collection since both the counter value and the corresponding wall clock estimate are recorded with each entry in the bet storage and the server has recorded the negotiated offsets.

If broadcast time stamps are available, they are usually utilized for adjusting local clocks in other systems. However, it is not secure to use the broadcast time stamps as they are typically used, e.g. in GPS, the NIST time service [18], and NTP, instead of resynchronizing over the two-way data link prior to the session. Because the limit h is stricter than g, Eq. (10) only fails before Eq. (4) if the complete broadcast transmission is shifted more than the amount of g or if a broadcast packet other than a tick packet is delayed outside $g + uT$. Therefore, if the protected counter, which is used for maintaining the wall clock estimate, has drifted too far from the server clock after the synchronization, it is already detected as a failure in Eq. (4). Without the protection against the cumulative delay, the broadcast time-stamps could be utilized for maintaining the wall clock synchronization for a benign user. Unfortunately, at the same time a malicious user could utilize the cumulative delay for postponing the end of the betting session. The cumulative delay would automatically make the adjusted wall clock estimate loose time by $g + uT$ per packet without detection, and thus, allow a successful attack.

The combined counter and wall clock verification methods detect all the possible attacks when only the protected counter rate or only the broadcast delay can be tampered with. The detection accuracy depends on the chosen timing parameters, the stability of the counter, and the accuracy of the knowledge of the utilized network technologies. Successful attacks can only be implemented if the attacker is able to modify the counter rate unlimitedly and delay the broadcast transmission simultaneously. Such an attack can technically be prevented by randomly reopening the two-way data link during the betting session, integrating the wireless network transceiver or, e.g., a reference GPS receiver inside the protected area. In that case, tampering with the counter rate also prevents the synchronization to the wireless medium, which is detected as a connection failure of the betting terminal. As a non-technical protection, the betting organizer should keep a log on winnings so that if a user starts succeeding suspiciously often, the authenticity of her activities can be verified.

6.3. Storage management

The accuracy and correctness of timing alone is not enough for ensuring the security of a betting session. It is also required that the stored bet records are authentic and their information, including the time stamps, has not been tampered with. An attack against the storage procedures is successful if a malicious user manages to fabricate, modify, or delete a bet record without detection.

In order to reduce the resource requirements and cost of the protected area, the storage for the session data is located in the unprotected area of the terminal. The storage entries are cryptographically chained in the protected area by the *storage management* entity. The chaining is based on [31] in which a method for creating protected audit logs in untrusted environments is designed. In this work the design is simplified as the stored data does not have to be kept secret (encrypted) from the user and there is no need for supporting partial access or verification of the stored data. In addition, in this work it is defined how the chain is bound to both the user and the protected area. The format of the storage contents are defined as well.

The chain is computed using a keyed Message Authentication Code (MAC) algorithm [32]. The chaining could also utilize digital signatures. However, a MAC algorithm requires less processing power, and thus, it is favored to decrease the processing requirements and cost of the protected area. The widely utilized HMAC [33] using SHA-256 [22] or Advanced Encryption Standard (AES) [34] in the CMAC mode [35][20] are well-suited for the purpose.

A storage entry consists of data and an appended chain value. The chain value is computed from the data, the chain value of the previous entry, and the current chain key A_i with the MAC algorithm

$$\text{chain}_i = \text{MAC}(A_i, \text{chain}_{i-1} | \text{data}_i). \tag{11}$$

The previous chain value for the first entry is chosen to be zero [31].

The initial chain key is generated from the master key K_m of the user authentication and from a unique seed q, which the protected area receives from the server in the storage setup of the registration. The parameter q can also be derived from the session key between the protected area and the server using the key separation as in Eq. (1). It is required that q is long, at least 128 bits, in order to thwart brute force searching for the initial chain key. Using the key separation, the initial chain key is

$$A_0 = H(H(\text{“StorageKey”} | K_m) | q). \tag{12}$$

K_m of the inner hash ties the identity of the user to the chain. On the contrary, since q is defined secretly from the user, it protects the chain against user modifications. After this, q is irretrievably destroyed at the terminal. The server computes the same initial key and stores it with the user registration data for the final verifications.

Thereafter, a different key is used for each stored entry [31]. After storing an entry, a new key is computed

$$A_{i+1} = H(\text{“UpdateKey”} | A_i). \tag{13}$$

The previous key is always irretrievably destroyed. Hence, even compromising a terminal will not allow changing entries that have been generated with a used key. Resetting or shutting down the protected area must also destroy

the chain key. This prevents a malicious user from attempting to turn back the protected counter through a restart, e.g., if the counter starts every power cycle from zero. If the chain key is lost, the user must reregister and resynchronize for the event to be able to continue the session.

The chain ensures the authenticity of the data, since altering or deleting an entry makes also the rest of the storage invalid. Since each entry is protected with a different key and the keys are generated with a cryptographically secure one-way hash function, an attacker can only attack a single entry alone. The amount of available data per key is minimized. Furthermore, since each key is used only once and each storage entry has a strictly defined format (next section), the attacker cannot exploit the protected area for requesting additional MACs for a key or for freely chosen data.

6.4. Stored data

To create a complete, verifiable trace for a betting session, a new entry is written to the data storage for each received broadcast packet and placed bet. Each entry generated by the storage management contains event ID, entry ID, size, type, protected counter value at the time of entry creation, protected wall clock estimate at the time of the entry creation, entry data, and chain value. The entry ID is a per-event sequence number for the storage entries maintained in the protected area. The value of the type field is *initial entry*, *broadcast packet*, *bet record*, or *final entry*. The contents of the entry types are presented in Fig. 12.

At the end of the registration, the storage management opens the storage by creating an initial entry with the initial chain key [31]. The entry data are user ID, terminal ID, and initial balance. The user and terminal ID are the same that were utilized in the user and terminal authentication. When the terminal receives a valid broadcast packet, the entry data copied from the packet are event ID, packet ID, packet type, and tick or bet ID (zero for a bet info packet). For a placed bet the entry data are bet ID, identifier of the chosen option, stake, and current balance. The identifiers are copied from an open bet packet. The stake is given by the user and the current balance is computed by the game engine. When a betting session ends or when the user decides to stop betting, the storage is sealed with the final entry.

Each stored entry has unique contents (unique event ID under which the entry IDs are unique) and defined format. In addition to the unique keys, this further restricts an attacker not knowing the chain key to attempting the modification of very limited number of entry fields. When a secure MAC algorithm is used, it is not possible to change those fields – especially into anything meaningful – without also changing the MAC value. The purpose of the initial and final entries is to prevent the user from restarting the storage or wiping out the complete session data [31]. After the initial entry has been created, data can only be added to the storage. After the final entry, the storage management destroys the chain key to disable further additions.

After collecting the session data, the server verifies that the storage chain is valid, based on the recorded registration data. It also checks that the time stamps and counter values of the entries are consistent with each other and with the broadcast timing information stored at the server during the session.

6.5. Violations

Since during an offline RTB session terminals only receive data over an unreliable broadcast link, it is probable that some packets are missed. Casually missing an open, close, result, or info packet is not crucial for the security of the organizer. If a bet packet is missed, a user misses the information related to a bet target. The missed information is received in the next bet info packet. The user only has less time for betting or receives the placement acceptance notification or the bet result later. A tick packet miss or failing in tick timing verifications at a terminal is considered more severe.

The protected area maintains two separate counters, *violation counter* and *severe violation counter*, for logging failures during a betting session. A bet or info packet miss or

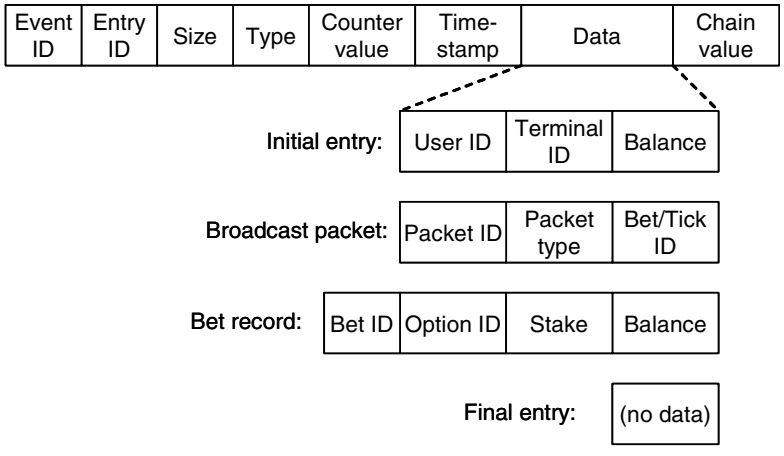

Fig. 12. Contents of stored entries.

timing error increment the violation counter. A tick packet miss or timing error increment the severe violation counter. The server defines thresholds for both the counters in the registration. If a threshold is reached, the data storage is sealed with the final entry and betting is disabled. The bets placed so far are valid. By defining the limits to one, any violation disables continuing betting.

7. Technologies for offline RTB

There are several alternatives for offline RTB implementation technologies. In set-top boxes DVB can be utilized for the broadcast data delivery whereas the two-way communications utilize an available return channel (e.g. IP network, analog modem). The set-top box carries out the unprotected processing and the protected area is implemented in an additional expansion module. Currently, a new standard, DVB-H, for handheld devices is emerging. It expands the DVB coverage from set-top boxes to small, portable devices with less processing capacity. For instance, a PDA or a mobile phone with GPRS/UMTS and DVB-H support is a suitable platform for the offline RTB. Most mobile phone producers have already built DVB-H terminals. Another solution for advanced multimedia terminals as well as small hand-held devices can utilize Digital Audio Broadcasting (DAB) as the broadcast technology.

As supporting both the broadcast and the two-way traffic, WLAN technology is suitable for localized RTB services, e.g. in a stadium. When WLAN terminals only passively receive broadcast traffic during a betting session, the limit for the maximum number of terminals in one network is unrestricted. Depending on the terminal type, the protected processing can be implemented on a PC-card or some other dedicated expansion card.

Currently, smart cards are utilized in a wide range of e-commerce applications for authentication and storing private information. Due to their tamper-resistance and support for cryptographic operations, they have suitable characteristics for the implementation of the protected area of the offline RTB terminal. The protected area control can be implemented in the smart card processor and the rest of the processing as special functional units. However, current smart cards do not contain fixed, internal counters for time-keeping. Instead, they support variable operating frequencies supplied by an external oscillator. It is also recommended that smart cards randomly vary their internal clock cycle length during the operation in order to prevent timing and power attacks against cryptographic operations [36]. These features make smart cards in their current form unsuitable for the offline RTB.

In addition to dedicated solutions, the technology developed by Trusted Computing Group (TCG) [37] has potential for the protected area implementation. TCG is defining hardware-based solutions in order to increase the level of security and trust in various computing platforms, such as PCs and mobile terminals. The devices are specified to include a tamper-resistant Trusted Platform Module (TPM) that provides cryptographic operations and verifies the integrity of the computing environment of the main platform. The TPM specification also includes a hardware counter that can be used for time-stamping with digital signatures. Of existing technologies, TPM is the closest match with the protected area design of this work. However, TPM does not define any method for ensuring the relationship (synchronization) between its counter and an external clock ([38], p. 114), which makes it inapplicable as such for offline RTB terminals. For the offline RTB, TPM should also provide a method for securely chaining the data (i.e. bet records) it has time-stamped and signed to prevent erasing them afterwards. The main purpose of TPM is to protect user data against software attacks. It has also been criticized for supporting digital rights management.

If smart cards or TPM included all the features required by the protected area, they would be perfect solutions for the offline RTB. Smart cards are already widely deployed and the TPM technology is emerging in the newest PCs and laptops. The authors believe that the features required by the offline RTB would be useful in other applications as well. They are especially beneficial in applications in which a choice made by a user and the time instant at which it was made are important.

8. Offline RTB prototype

In order to test offline RTB in practice, prototypes for WLAN and DVB environments have been implemented [39]. They enable evaluating design alternatives, network technologies, UIs, and end-user behavior in real environments. The prototypes have been experimented by several test groups. Valuable information on the attractiveness of the concept as well as its usability and applicability has been collected. Main aspects have been to investigate how near the closing time bets are placed, what kinds of bets are suitable, and what a suitable interval for announcing bets is. From the technical point of view, user behavior places requirements on the system performance. The current WLAN implementation can be directly applied for providing localized betting services. It has been tested in an ice-hockey stadium.

8.1. Implementation

The offline RTB prototype developed for WLAN environment is illustrated in Fig. 13. The main functional parts of the terminals and the betting server have been implemented in SDL. SDL is connected to the environment through user and network interfaces. The implementation does not limit the number of terminals accessing the service and it supports several operators and parallel events as well as parallel bets.

The prototype consists of a betting server, one or more operator terminals, an odds server, and several betting terminals. The server can be located in a place with the best network capacity and physical security whereas the

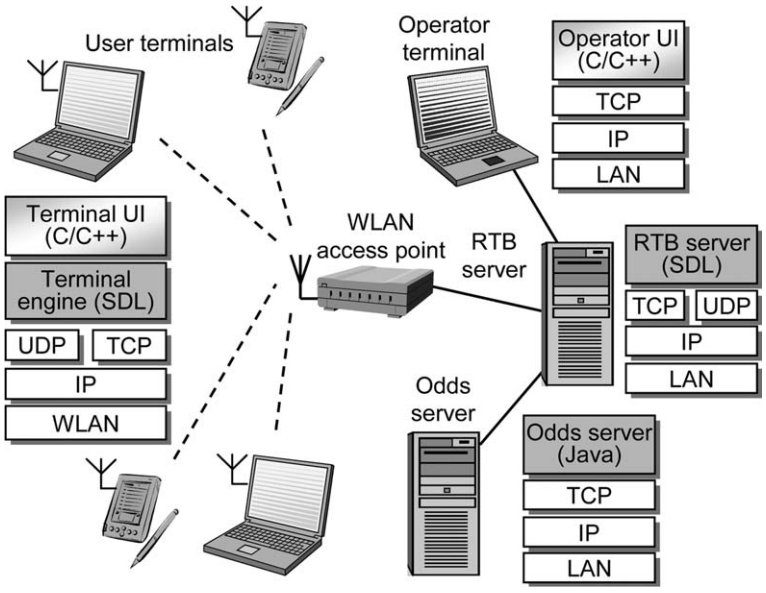

Fig. 13. Offline RTB prototype in WLAN environment.

operator in a suitable place for following the event. The server and operator software run in Windows XP environment. The betting terminal is implemented for Windows XP, Pocket PC, and DVB set-top boxes. The prototype utilizes UDP/IP for the broadcast and TCP/IP for the two-way data connection. The broadcast link for set-top boxes is DVB. An additional PC is used for receiving the WLAN broadcast and converting it into the DVB broadcast.

The operator uses predefined bet templates, which can be used as such or the texts and odds can be modified. The operator can also employ a separate server for dynamic odds. The odds are automatically adjusted throughout an event, utilizing game time, current scores, and expected final scores. When an operator announces a bet with dynamic odds, the offline RTB server transmits a request to the odds server. After the reply, the server replaces the odds provided in the template and broadcasts the announcement to the terminals.

8.2. User interfaces

Suitable UIs for both the RTB users and operators have been evaluated using the prototype. The UIs have gone through several generations and various improvements as well as optional features have been developed, based on the feedback of real end-users.

In general, the usability tests with different implementations have shown that the displayed information must be very clear and concise. Otherwise betting disturbs following the event instead of adding entertaining value to it. Also, because the time for placing a bet is strictly limited, compact and comprehensible information leaves more time for making the actual decision and increases the usability. Incidents should preferably have only few (two or three) outcome alternatives in order to make the information effortlessly manageable for the users as well as for the operators. It has also turned out that during a betting session the users prefer making all selections with the minimal effort, i.e., with a single keystroke or tap. According to the feedback, the RTB service has been found inspiring and entertaining.

Fig. 14a shows the betting UI of the RTB prototype for the Pocket PC touch screen and Fig. 14b for a set-top box in their current forms. Because current set-top box remote controls are fairly slow to use compared to a touch screen, the set-top box UI has been made simpler. The set-top box UI is semi-transparent and it can be hidden to free the full screen for the video. In addition to making a choice, the UIs allow the user to change the stake, discard a bet, view previous bets and their results, and finish the betting session.

When a bet is opened or a bet result received, the UIs show the information immediately on the screen. If the set-top box UI is hidden, only a small notification symbol is shown on a corner of the screen. An optional sound notification can be generated in both UIs so that the user does not frequently have to check the screen while waiting for announcements. Vibration can also be used as another way to notify the user in portable betting terminals, especially in noisy environments (e.g. stadiums). As the bet text is large, the user can quickly decide if the shown information is interesting. In the Pocket PC a bet is placed by simply tapping on one of the provided options. In the set-top box the choice is made with the remote control. The same stake is used for the bets as long as the user changes it. The user can also turn on optional confirmations to prevent accidental placements.

According to the usability experiments, even though the betting screen must be concise by default, the users should

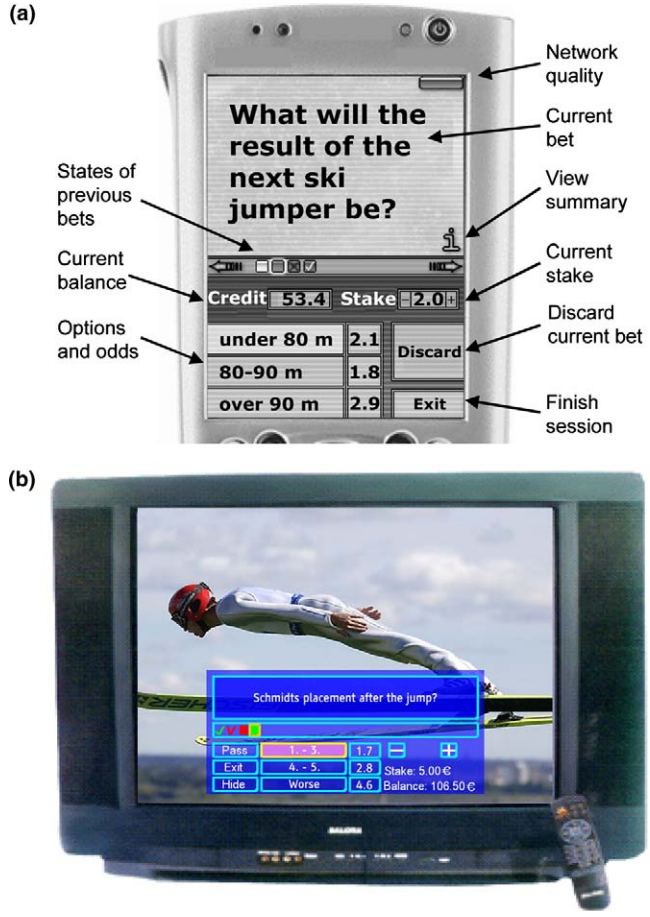

Fig. 14. Offline RTB user interfaces for: (a) Pocket PC and (b) set-top box.

also be able to get more information when they desire. Specifically, it should be easy to browse through previous bets. For the purpose, the UIs maintain lists of passed bets. The user always sees the states of previous bets (open, closed, placed, waiting result, won, or lost) represented as symbols below the bet text. She can check the information of a specific bet by selecting its symbol. Uninteresting bets can be removed from the list. In addition, the user can view a session summary and more detailed information on the bets by choosing the summary view.

In addition to the Pocket PC and set-top box UIs, a stadium screen interface has been developed and tested with the RTB prototype. The bet announcements are shown on the stadium screen, which releases the users from frequently checking their personal betting terminals in the noisy and crowded environment. The operator can choose whether the bet information occupies the whole screen or only parts of it. Similarly to the set-top box UI, the notification can be a small blinking logo or a sound and the detailed bet information is shown on the user terminals. On the other hand, when the stadium screen is utilized for providing the full information, the terminal UI can be made simpler. For example, the UI can consist of a small screen and few buttons just for making a choice.

The main screen of the operator UI is presented in Fig. 15. On the top of the window the operator starts and finishes betting sessions. The bets for a session are managed using the left half of the UI. As the states of the bets are changed, the bets flow from the top to the bottom of the UI. A new bet is defined using empty or prefilled templates as well as static or dynamic odds. The right side of the UI is utilized for reporting real-time session data (number of bets, registered users etc.). The operator can also broadcast textual information to the user terminals and choose the view for the terminals supporting multiple views (e.g. the stadium screen).

The UI tests have shown that the operator should not announce bets too frequently, again, not to interrupt following the event itself. On the other hand, limiting the announcement frequency for all the users in the same way is too restricting. Some users want to have more intense betting sessions than others. Thus, the RTB prototype is being further developed to support different bet announcement patterns. The users will be able to choose the intensity of their betting sessions and also to change their preferences during the sessions. The intensity level affects the announcement frequency and the time to make a decision. The bets are categorized according to their characteristics and the UI chooses which announcements to display based on the chosen level.

8.3. User behavior

In addition to the usability tests, the offline RTB prototype has been utilized for collecting data related to the timing of bet placements during real betting sessions. The target events for the experiments have been ski jumping and ice-hockey, which have different characteristics related to RTB. Ski jumping is strictly periodic with predictable closing times whereas in ice-hockey it is difficult to distinguish discrete incidents. The tests presented here were arranged for two small groups of volunteers, the first one had 21 members and the second one seven. The ski jumping test was performed for both the groups and the ice-hockey test for the first group only.

The results of the timing behavior experiments are shown in Fig. 16. The graphs present how long a bet was open and how near the closing times users placed their bets on average. At the beginning of the ski jumping session, the behavior of new users follows closely the opening times. However, as the users become more familiar with the betting, they start delaying placements nearer the predictable closing time. On the contrary, in ice-hockey the placement times follow the opening times throughout the session because of the unpredictable closing times. It was also verified that defining an estimate for the closing times changes the user behavior in the second test closer to the first one.

In Fig. 16a, the bets 5–11 are especially interesting. The users made their decisions about 5 s before the closing time on average, even though bets were open up to 27 s. This supports the performance simulations and analysis pre-

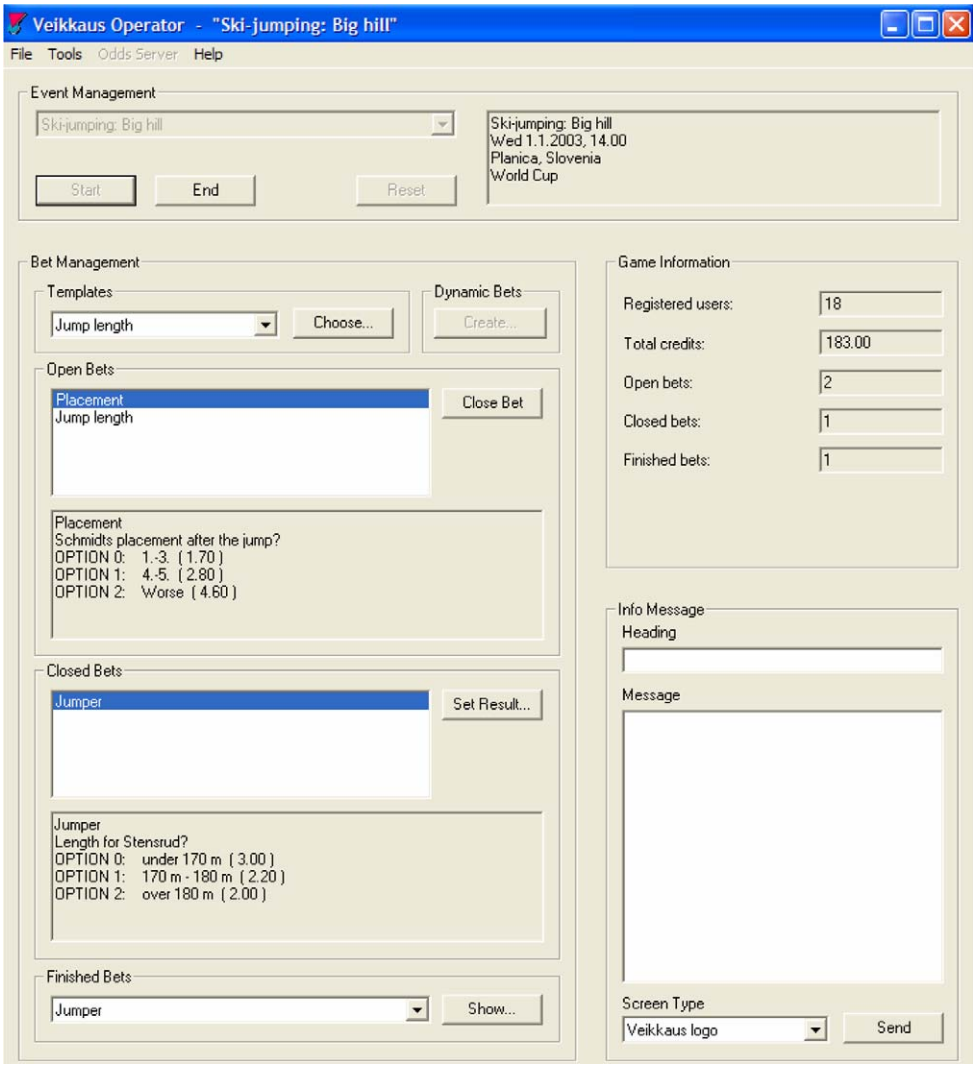

Fig. 15. Offline RTB prototype user interface for operator.

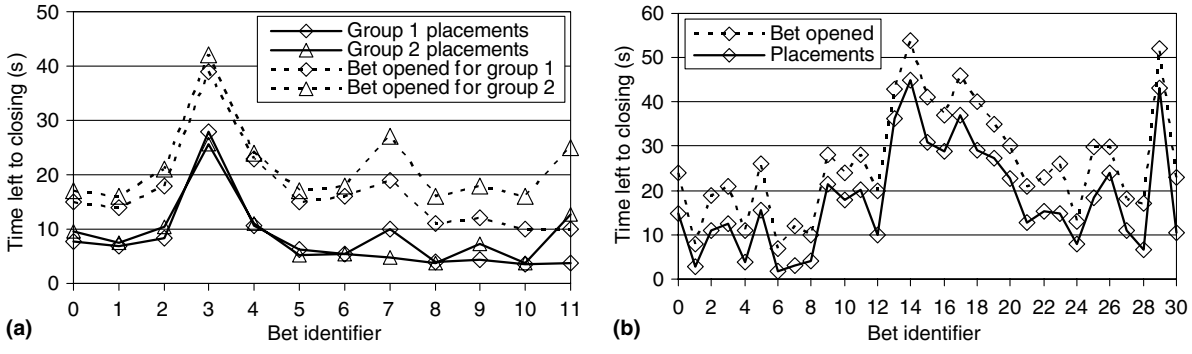

Fig. 16. Results of the timing experiments with the offline RTB prototype: (a) ski jumping and (b) ice-hockey.

sented in Section 4. Another interesting aspect is that even though both the test groups attended the RTB sessions for the first time and without any prior knowledge or guidance, the anticipated timing behavior started to show already after the first four bets.

9. Conclusions

The rapid development of communication systems as well as e-commerce technologies has enabled electronic betting to become an important field of e-commerce and

entertainment. However, the fundamental nature of betting itself has not changed. There are not enough processing resources in electronic betting systems for changing the services into interactive sessions with full scalability support. This was also shown in the simulations and experiments of this work.

In order to provide large-scale electronic betting as a new type of interactive, real-time entertainment, a new approach was presented in this paper, referred to as offline RTB. The offline architecture enables betting in short time cycles while keeping processing requirements low. It results in new challenges for security and timing as the operation environment of a betting terminal has to be considered hostile. Thus, reliable methods for verifying the timing and the authenticity of the terminal operations were developed. In addition to the reliable protocols and algorithms, the critical operations are implemented in a protected area which the user of the betting terminal cannot access.

Regardless of the dependability of the technical protection, the offline RTB still requires agreed rules, which can be applied if malpractices are detected. A trusted supervisor is required for ensuring the authenticity of the organizer operations. Limiting the stakes makes the service less appealing for break attempts with financial goals. Also, instead of providing bets on which large sums would be invested, the goal of the offline RTB is to offer frequent, short bets with small sums. This way the service can be applied for adding entertaining value to various events. In addition, the design is applicable for other types of contests without monetary goals.

References

[1] European Game and Entertainment Technology Ltd Ab (EGET) website, 2005. Available from: <http://www.eget.fi>.

[2] Global Interactive Gaming website, 2005. Available from: <http://www.gigltd.com>.

[3] E. Kushilevitz, T. Rabin, Fair e-lotteries and e-casinos, in: Proceedings of the Cryptographer's Track at RSA Conference 2001 (CT-RSA 2001), San Francisco, USA, Apr., 2001, pp. 100–109.

[4] C. Hall, B. Schneier, Remote electronic gambling, in: Proceedings of the 13th Annual Computer Security Applications Conference (ACSAC '97), San Diego, USA, Dec., 1997, pp. 26–29.

[5] M. Jakobsson, D. Pointcheval, A. Young, Secure mobile gambling, in: Proceedings of the Cryptographer's Track at RSA Conference 2001 (CT-RSA 2001), San Francisco, USA, Apr., 2001, pp. 110–125.

[6] W. Aiello, A. Rubin, M. Strauss, Using smartcards to secure a personalized gambling device, in: Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS '99), Singapore, Nov., 1999, pp. 8–12.

[7] W. Zhao, V. Varadharajan, Y. Mu, Fair on-line gambling, in: Proceedings of the Annual Computer Security Applications Conference (ACSAC 2000), New Orleans, USA, Dec., 2000, pp. 394–400.

[8] K. Sako, Implementation of a digital lottery server on WWW, in: Proceedings of Secure Networking – CQRE (Secure) '99, Düsseldorf, Germany, Nov.-Dec., 1999, pp. 101–108.

[9] J. Zhou, C. Tan, Playing lottery on the Internet, in: Proceedings of the 3rd International Conference on Information and Communications Security (ICICS 2001), Xian, China, Nov., 2001, pp. 189–201.

[10] P.A. Fouque, G. Poupard, J. Stern, Sharing decryption in the context of voting and lotteries, in: Proceedings of the 4th International Conference on Financial Cryptography (FC 2000), Anguilla, British West Indies, Feb., 2000, pp. 90–104.

[11] P. Syverson, Weakly secret bit commitment: applications to lotteries and fair exchange, in: Proceedings of Computer Security Foundations Workshop (CSFW'98), Rockport, USA, Jun., 1998, pp. 2–13.

[12] S. Haber, W.S. Stornetta, How to time-stamp a digital document, J. Cryptol. 3 (4) (1991) 99–111.

[13] Symmetricom, Inc. website, 2003. Available from: <http://www.symmetricom.com>.

[14] SDL Forum Society website, 2005. Available from: <http://www.sdl-forum.org>.

[15] IEEE Std 802.11b-1999, Wireless LAN medium access control (MAC) and physical layer (PHY) specifications: higher-speed physical layer extension in the 2.4 GHz band, 1999.

[16] H.S. Wang, N. Moayeri, Finite state Markov channel – a useful model for radio communication channels, IEEE Trans. Vehicular Technol. 44 (1) (1995) 163–171.

[17] S. Karande, S.A. Khayam, M. Krappel, H. Radha, Analysis and modeling of errors at the 802.11b link layers, in: Proceedings of 2003 IEEE International Conference on Multimedia and Expo (ICME 2003), Jul., 2003, Baltimore, USA, vol. I, pp. 673–676.

[18] National Institute of Standard and Technology (NIST) Time & Frequency Division website, 2005. Available from: <http://tf.nist.gov>.

[19] D. Mills, Network time protocol (version 3) specification, implementation and analysis, RFC 1305, 1992.

[20] T. Iwata, K. Kurosawa, Stronger security bounds for OMAC, TMAC and XCBC, National Institute of Standards and Security (NIST), 2005. Available from: <http://csrc.nist.gov/CryptoToolkit/modes/comments/>.

[21] T. Dierks, C. Allen, The TLS protocol 1.0, RFC 2246, 1999.

[22] Federal information processing standards publication (FIPS) 180-2, Secure hash standard, 1999.

[23] D. Taylor, T. Wu, N. Mavrogiannopoulos, T. Perrin, Using SRP for TLS authentication, TLS Working Group Internet Draft, 2005.

[24] Kerberos website, 2005. Available from: <http://web.mit.edu/kerberos/www/>.

[25] Federal information processing standards publication (FIPS) 186, Digital signature standard (DSS), 1994.

[26] Taitien Electronics website, 2005. Available from: <http://www.taitien.com>.

[27] D. Mills, Simple network time protocol (SNTP) version 4 for IPv4, IPv6 and OSI, RFC 2030, 1996.

[28] IEEE Std 1588-2002, IEEE standard for a precision clock synchronization protocol for networked measurement and control systems, 2002.

[29] Network Time Synchronization Project website, 2005. Available from: <http://www.eecis.udel.edu/~mills/ntp.html>.

[30] F. Cristian, A probabilistic approach to distributed clock synchronization, in: Proceedings of the 9th International Conference on Distributed Computing Systems, Newport Beach, USA, June, 1989, pp. 288-296.

[31] B. Schneier, J. Kelsey, Cryptographic support for secure logs on untrusted machines, in: Proceedings of the 7th USENIX Security Symposium, San Antonio, USA, Jan., 1998, pp. 53-62.

[32] A.J. Menezes, P.C. Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.

[33] Federal information processing standards publication (FIPS) 198, The keyed-hash message authentication code, 2002.

[34] Federal information processing standards publication (FIPS) 197, Advanced encryption standard (AES), 2001.

[35] M. Dworkin, Recommendation for block cipher modes of operations: the CMAC mode for authentication, National Institute of Standards and Technology (NIST) special publication 800-38B, 2005.

[36] O. Kömmerling and M.G. Kuhn, Design principles for tamper-resistant smartcard processors, in: Proceedings of USENIX Workshop on Smartcard Technology (Smartcard '99), Chicago, USA, May, 1999, pp. 9-20.

[37] Trusted Computing Group (TCG) website, 2005. Available from: <http://www.trustedcomputinggroup.org>.
[38] Trusted Computing Group (TCG), TPM main specification – Part 2: TPM Structures, version 1.2, level 2, revision 85, February 2005.
[39] P. Hämäläinen, M. Hännikäinen, T.D. Hämäläinen, and R. Soininen, Offline architecture for real-time betting, in: Proceedings of 2003 IEEE International Conference on Multimedia and Expo (ICME 2003), Baltimore, USA, Jul., 2003, vol. I, pp. 709–712.